UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Henrique Fortes Raia

NAVEGAÇÃO DE ROBÔS MÓVEIS COM RESTRIÇÕES TEMPORAIS USANDO ÁRVORES DE COMPORTAMENTO E AÇÕES CORRETIVAS

Florianópolis (SC)
2016

Henrique Fortes Raia

NAVEGAÇÃO DE ROBÔS MÓVEIS COM RESTRIÇÕES TEMPORAIS USANDO ÁRVORES DE COMPORTAMENTO E AÇÕES CORRETIVAS

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação para a obtenção do grau de "Mestre em Ciência da Computação".

Orientadora:
Prof[a]. Dr[a]. Patricia Della Méa Plentz, UFSC.

Florianópolis (SC)
2016

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Raia, Henrique Fortes
Navegação de Robôs Móveis com Restrições Temporais Usando Árvores de Comportamento e Ações Corretivas / Henrique Fortes Raia ; orientadora, Patricia Della Méa Plentz - Florianópolis, SC, 2016.
86 p.

Dissertação (mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico. Programa de Pós-Graduação em Ciência da Computação.

Inclui referências

1. Ciência da Computação. 2. Previsão de perda de deadline. 3. Grupo de robôs móveis. 4. Ações corretivas. 5. Árvore de Comportamento. I. Plentz, Patricia Della Méa . II. Universidade Federal de Santa Catarina. Programa de Pós Graduação em Ciência da Computação. III. Título.

Dedico este trabalho aos meus pais que sempre me apoiaram e me incentivaram.

Agradecimentos

Agradeço primeiramente a Deus por ter me dado saúde e força para superar as dificuldades.

Gostaria de agradecer, também, aos meus pais Edson Luiz Raia e Claudia Fátima Fortes Raia, bem como à minha irmã Amanda Fortes Raia e à minha namorada Carla Mendes da Silva por todo o apoio e incentivo que têm me dado durante esse tempo que ficamos afastados.

Sou muito grato, também, à minha orientadora, a professora Patricia Della Méa Plentz pela paciência e por todo o auxílio que me prestou durante o mestrado. Sem sua orientação, este trabalho não teria alcançado os resultados que obteve. Agradeço ao professor Edson Roberto de Pieri, à professora Luciana de Oliveira Rech e à professora Carla Merkle Westphall por terem aceito o convite de participar na banca examinadora deste trabalho. Suas contribuições auxiliaram a enriquecer o conteúdo aqui apresentado. Agradeço também à Katiana de Castro Silva, por ter sido paciente comigo e me ajudado sempre que precisasse de auxílio.

Também agradeço aos meus amigos, que sempre me auxiliaram. Dentre eles gostaria de agradecer principalmente a Sidney Roberto Dias de Carvalho, Michel Platini Gomes da Silva, Ruan Carlos Ramos da Silva e Lucas Adriano de Matos Vilhagra. Sou grato ao primeiro por ter me feito companhia durante minha estadia em Florianópolis, sempre disposto a conversar sobre aleatoriedades enquanto tomávamos uma Coca-Cola. Agradeço aos outros três pelas reuniões online que tínhamos semanalmente, nas quais sempre nos divertíamos muito. Sou grato ao Luís Fernando Arcaro pelos convites para participar de trilhas, pela companhia nas fritas com bacon e pelo auxílio ao fim deste trabalho. Agradeço ao Fernando de Lucca Siqueira por suas contribuições a este trabalho, pela ajuda na escrita do artigo e por me auxiliar a compreender o conceito de árvore de comportamento e como utilizar a biblioteca *behavior_trees* para o ROS.

Por fim, gostaria de agradecer a todos que, direta ou indireta-

mente, contribuíram para a conclusão deste trabalho.

"Ter consciência da própria ignorância já é um passo para o saber."

– Benjamin Disraeli

Resumo

A cada dia, o número de robôs móveis usados para realizar diferentes tarefas aumenta. A popularização da robótica móvel nos últimos anos se deve à redução de custo de hardware e ao crescimento e melhora da infraestrutura de software. Muitas das tarefas realizadas pelos robôs móveis possuem restrições temporais. Em alguns casos, torna-se interessante ter uma previsão precisa a respeito do cumprimento da tarefa no prazo estipulado. Quando considera-se um grupo de robôs executando uma tarefa coletiva, a previsão do não cumprimento de uma tarefa permite que ações sejam realizadas para minimizar o prejuízo. Este trabalho apresenta um conjunto de ações corretivas para serem realizadas em tarefas que consistem na navegação de um robô móvel entre um ponto origem e um ponto objetivo. Para a tomada de decisão, uma abordagem utilizando o conceito de Árvore de Comportamento foi aplicada. Devido à utilização das ações corretivas, os robôs foram capazes de cumprir as restrições temporais das tarefas em mais iterações do que quando estas ações estão desabilitadas. Os robôs são, também, mais eficientes na execução das tarefas, conseguindo um desempenho superior quando utilizavam as ações corretivas.

Palavras-chave: Previsão de perda de *deadline*, Grupo de robôs móveis, Ações corretivas, Árvore de Comportamento.

Abstract

At each day, the number of mobile robots, used to perform different tasks, increases. The popularization of mobile robotics in the last years was caused by the reduction of the cost of hardware and the growth and improvement of software infrastructure. Most of the tasks performed by mobile robots have temporal restrictions. In some cases, it is interesting to have an accurate prediction regarding the task accomplishment in the given deadline. When a group of robots is considered to perform a collective task, the prediction of a non-fulfillment of a task within its temporal restriction allows that actions can be performed to minimize losses. This work presents a set of recovery actions to be performed in tasks that consist on the navigation of a mobile robot between a starting point and a goal point. For the decision-making process, an approach using the concept of Behavior Tree was used. Due the utilization of the recovery actions, the robots were able to fulfill the temporal constraints of the tasks in more iterations than when these actions were unavailable. The robots were, also, more efficient when executing the tasks, achieving a superior performance using the recovery actions.

Keywords: Deadline missing prediction, Group of mobile robot, Recovery actions, Behavior Tree.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

Lista de Símbolos

Sumário

Capítulo 1

Introdução

Este trabalho visa o desenvolvimento de ações corretivas a serem executadas quando um robô móvel prevê perda de *deadline*. Estas ações têm o intuito de auxiliar o robô a cumprir as restrições temporais da tarefa ou de reduzir o prejuízo causado pela perda do prazo estipulado para a tarefa. A seção 1.1 apresenta uma contextualização sobre o problema a ser tratado nesta dissertação. O objetivo geral deste trabalho é explicado na seção 1.2 e os objetivos específicos estão discriminados na seção 1.3.

1.1. Contextualização

A cada dia, o número de robôs móveis, utilizados para realizar as mais diversas tarefas, aumenta. A redução do custo de hardware é um dos principais fatores que incentivou este aumento. Robôs móveis podem ser aplicados em diversas tarefas e são de grande importância quando a tarefa pode apresentar algum risco ao ser humano.

Conforme apresentado na literatura, os robôs móveis são bastante usados na exploração de ambientes [JIA et al., 2004; MEI et al., 2006], na perseguição de alvos móveis [ZHU et al., 2013] e no deslocamento de carga de um ponto a outro do ambiente [CAI et al., 2014], entre outras atividades. Na maioria dos casos, a tarefa de deslocamento do robô pode ser limitada por restrições temporais. Como exemplo pode-se citar um conjunto de veículos autônomos utilizados para carga ou descarga de produtos em um centro de distribuição. O atraso para descarregar caminhões ou para carregá-los com produtos pode gerar prejuízos financeiros para a empresa responsável por esse centro de distribuição.

Prever se as restrições temporais, impostas para a navegação

de um robô em um determinado ambiente, serão respeitadas é essencial. Tendo este conhecimento, ações podem ser tomadas para que o prazo estipulado seja cumprido ou para que os prejuízos relacionados com a perda do *deadline* sejam minimizados. Esta questão de pesquisa torna-se ainda mais interessante quando considera-se um grupo de robôs móveis realizando uma tarefa colaborativa. Os robôs móveis podem ser considerados Sistemas de Tempo Real (*Real-Time Systems* ou RTS), ou seja, sistemas nos quais as tarefas realizadas estão sujeitas a restrições temporais.

Para que um sistema de tempo real apresente comportamento correto, é necessário que os resultados obtidos sejam corretos e que respeitem as restrições temporais impostas a esse sistema. Caso o *deadline* do sistema não seja respeitado, os benefícios trazidos podem ser reduzidos, anulados ou, dependendo da aplicação, o não cumprimento das restrições temporais pode acarretar em falhas críticas no sistema. Os sistemas de tempo real podem ser classificados de acordo com sua segurança [FARINES et al., 2000]:

- Sistemas não críticos de tempo real (*soft real-time systems*) são sistemas nos quais a perda de um *deadline* acarreta uma redução nos benefícios desse sistema que é proporcional ao tempo de resposta;
- Sistemas críticos de tempo real (*hard real-time systems*) são sistemas nos quais o não cumprimento de uma restrição temporal leva à falha do sistema;
- Sistemas de tempo real firmes (*firm real-time systems*) são sistemas nos quais a perda de um *deadline* não leva a uma falha do sistema, mas essa perda reduz os benefícios do sistema.

Existem diversos algoritmos e abordagens de navegação para robôs móveis. Alguns algoritmos usam informações de sensores, enquanto outros utilizam apenas informações do ambiente em que o robô navega. Há, também, abordagens híbridas, que reúnem características de duas ou mais técnicas. Recentemente, o conceito de Árvores de Comportamento (*Behavior Trees* ou BT) vem sendo aplicado em sistemas robóticos [BAGNELL et al., 2012].

Árvores de Comportamento são modelos matemáticos para descrever planos de execução e o comportamento do sistema por meio de uma estrutura modular e hierárquica. A Árvore de Comportamento é uma poderosa ferramenta que pode ser usada para modelar o comportamento e controlar o processo de tomada de decisão na robótica. Há diversas vantagens na utilização deste modelo, quando comparado a modelos anteriores, tais como Máquinas de Estado Finito (*Finite*

State Machines ou FSM) ou Máquinas de Estado Finito Hierárquicas. As principais vantagens que podem ser citadas são modularidade, legibilidade e reusabilidade. Elas podem também melhorar a segurança, eficiência e robustez do sistema.

Neste trabalho, um conjunto de robôs móveis navegam de um ponto a outro (ponto origem e ponto destino). A tarefa que realizam durante a navegação consiste no transporte de objetos. Os robôs navegam, em momentos diferentes, até o ponto destino onde recebem alguns objetos para serem carregados para o ponto origem. Antes de retomar a navegação para o ponto origem, o mecanismo de previsão é acionado, o qual determina uma probabilidade do robô cumprir o deadline da tarefa. Este trabalho propõe ações corretivas, que serão realizadas quando o mecanismo de previsão determina uma probabilidade de perda de deadline. Estas ações irão, portanto, auxiliar o robô a realizar a tarefa no prazo definido (*deadline*).

1.2. Objetivo Geral

Desenvolver ações corretivas para serem executadas quando um robô móvel prevê uma perda de *deadline* da tarefa que está executando. Estas ações corretivas permitem que o robô termine a tarefa, respeitando o seu *deadline*, ou reduzir os prejuízos que podem ser causados pelo não cumprimento de suas restrições temporais.

1.3. Objetivos Específicos

Visando alcançar os objetivos gerais propostos, foram definidos alguns objetivos específicos:

1. Adaptar o mecanismo proposto por Monteiro [2014] para o *Robot Operating System* (ROS);
2. Implementar as ações corretivas utilizando o conceito de Árvore de Comportamento;
3. Aplicar, testar/validar e analisar o conjunto de ações corretivas desenvolvido;

1.4. Organização do Texto

O restante deste documento está dividido da seguinte forma: No capítulo 2 o mecanismo proposto por [MONTEIRO, 2014] é descrito com maiores detalhes. No capítulo também é apresentada a abordagem proposta. Os sistemas e ferramentas utilizados no desenvolvimento do

algoritmo são mostrados no capítulo 3. Uma descrição sobre a execução das simulações, bem como os cenários de simulação e os resultados também são apresentados no capítulo 3. No capítulo 4 é apresentada a revisão sistemática da literatura, além de trabalhos com temas relacionados ao tratado neste documento. No capítulo 5, as considerações finais, referentes ao trabalho apresentado neste documento, e os trabalhos futuros, são apresentados.

Capítulo 2

Árvore de Comportamento para Implementar Ações Corretivas em Robôs Móveis

Este capítulo está dividido da seguinte forma: na seção 2.1 é apresentado o conceito da Árvore de Comportamento. Na seção 2.2 é apresentado o mecanismo de perda de *deadline* proposto por Monteiro [2014], que é a base do trabalho apresentado neste documento. As alterações realizadas no mecanismo de Monteiro são mostradas na seção 2.3. A seção 2.4 discorre sobre o trabalho proposto neste documento. Uma conclusão referente aos assuntos abordados, apresentada na seção 2.5, encerra este capítulo.

2.1. Árvore de Comportamento

Sistemas robóticos convencionais usam Máquinas de Estado Finito (*Finite State Machines*, em inglês ou *FSM*) para modelar o estado da aplicação e controlar o comportamento do robô [MARINO et al., 2009]. A *FSM* permite que cada estado da aplicação, e as condições para que cada transição entre estados seja ativada, seja modelado. O principal problema com esta abordagem é que, para cada novo estado da aplicação, o modelo precisa criar novas transições entre todos os estados que podem culminar neste novo estado ou que podem ser alcançados a partir dele. Em uma aplicação cujo ambiente é dinâmico e que pode apresentar novos estados, a *FSM* pode ser bastante complexa. Uma abordagem mais recente e popular para modelar e controlar os estados de uma aplicação é a Árvore de Comportamento.

Árvores de Comportamento (do inglês, *Behavior Trees* ou *BT*) foram inicialmente utilizadas em jogos digitais para controlar a inte-

ligência artificial de personagens de jogos [ISLA, 2005]. Uma Árvore de Comportamento apresenta algumas vantagens em relação às máquinas de estado finito no que diz respeito à modularidade, legibilidade e reusabilidade. Em *FSM*s, o estado e as transições são atômicos e não armazenam informações sobre precedência, ou seja, a partir de qual estado ou a partir de qual transição resultou no estado atual. Por outro lado, como uma *BT* tem uma arquitetura de árvore, cada estado é uma evolução de um anterior. Marzinotto et al. [2014] define uma Árvore de Comportamento como um grafo acíclico com nodos e folhas, onde qualquer nodo sem um "filho" é considerado um nodo folha e o nodo sem "pai" é um nodo raiz.

A Árvore de Comportamento é composta por dois tipos de nodos: controle e execução. Nodos de controle são responsáveis por controlar a sequência de execução de uma Árvore de Comportamento. Um nodo de controle pode ser um nodo de sequência, representado por uma seta apontando à direita (→), ou um nodo seletor, representado por um ponto de interrogação (?). Um nodo de execução é um nodo folha que pode ser uma ação (representado por uma figura retangular) ou um nodo de condição (representado por um círculo). A figura 2.1 mostra os tipos de nodos e suas respectivas representações.

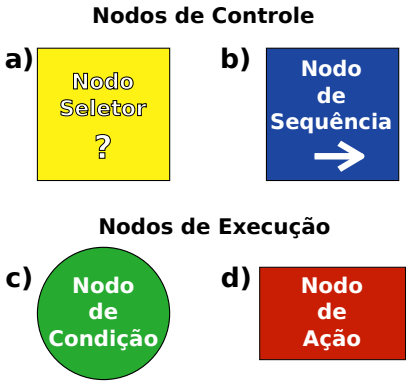


Figura 2.1: Tipos de nodos: a) Nodo Seletor b) Nodo de Sequência c) Nodo de Condição e d) Nodo de Ação

A execução de uma Árvore de Comportamento é baseada no retorno (*feedback*) dos nodos de execução. Periodicamente, a Árvore de Comportamento gera um sinal que é propagado do nodo raiz aos ramos da árvore, respeitando os nodos de controle. Cada nodo de execução, ao receber o sinal, irá executar e retornar o resultado de sua execução

para o nodo "pai". Quando recebe o sinal, o nodo de condição verifica o resultado de uma condição e retorna **sucesso** caso ela seja verdadeira e **falha** se for falsa. Um nodo de ação tentará executar uma ação ao receber o sinal. Esta ação pode ser qualquer função de uma aplicação. O nodo de ação retorna **sucesso** se a ação for concluída com êxito, **falha** caso ela não tenha obtido sucesso ou **executando**, caso ainda esteja sendo executada. O sinal para quando retorna ao nodo raiz.

O retorno dos nodos de controle dependerá do resultado de seus nodos filhos e do tipo de nodo de controle. Um nodo de sequência é um tipo de nodo que tentará executar todos os seus filhos, iniciando a execução a partir do filho mais à esquerda. Caso algum filho retorne sucesso, o nodo de sequência tentará executar o próximo filho. Se um filho retornar falha, o nodo de sequência irá encerrar sua execução e retornará um sinal de falha para seu nodo pai. Se um filho retornar que ainda está em execução, então este também será o retorno do nodo de sequência. O nodo de sequência apenas retornará sucesso se todos os seus filhos obtiverem sucesso em suas execuções. Este nodo pode ser interpretado como o operador **AND** em linguagens de programação. O nodo seletor tentará executar todos os filhos até que algum deles retorne sucesso. O primeiro filho que retornar sucesso irá encerrar a execução do nodo seletor que, por sua vez, retornará sucesso para seu nodo pai. Caso o nodo filho retorne que ainda está em execução, então o nodo seletor também retornará esse *status*. Ele apenas retornará falha no caso de todos os seus nodos filhos também retornarem falha. O nodo seletor pode ser comparado ao operador **OR** em linguagens de programação. As Figuras 2.2 e 2.3 resumem a execução dos nodos de sequência e seletor, respectivamente. Nestas figuras, o lado esquerdo mostra o estado de sucesso de cada nodo de controle e o lado direito mostra o estado de falha.

Como o sinal enviado da raiz para os ramos da árvore se propaga da esquerda para a direita, os nodos mais à esquerda da Árvore de Comportamento serão executados primeiro, indicando que quanto mais à esquerda um nodo está, maior é sua prioridade.

2.2. Mecanismo de Previsão de Perda de *Deadline* Proposto por Monteiro [2014]

Monteiro [2014] desenvolveu um mecanismo de previsão de perda de *deadline* que permite que um robô móvel seja capaz de prever se conseguirão completar uma tarefa dentro de um prazo. Este mecanismo pode ser executado tanto em ambientes estáticos quanto dinâmicos,

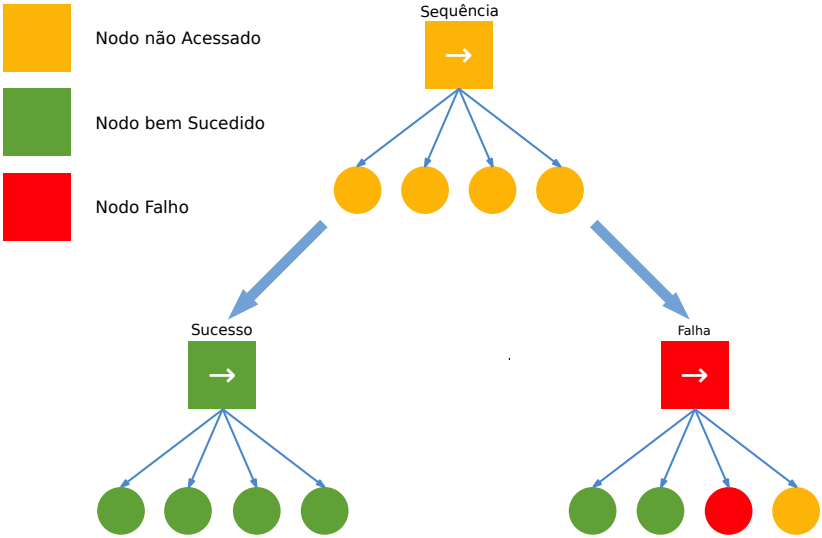


Figura 2.2: As condições de sucesso (à esquerda) e falha (à direita) para nodos de sequência (adaptado de [SIQUEIRA; PIERI, 2015])

armazenando em um histórico as posições dos obstáculos encontrados para que, caso se depare com a mesma situação, seja possível utilizar as informações contidas no histórico para auxiliar na previsão de perda de *deadline*.

O modelo de tarefas em que o mecanismo é aplicado consiste no robô deslocar-se de um ponto inicial até um objetivo e, então, retornar ao ponto inicial. Os tipos de ambiente considerados são os parcialmente conhecidos e os totalmente conhecidos. A cada execução da tarefa dada, o mecanismo armazena as informações obtidas, durante a navegação no ambiente, em um arquivo de histórico e utiliza estas informações no processo de previsão.

2.2.1. Histórico Principal

A primeira parte do processo de previsão é a construção de um histórico principal baseado na execução das tarefas. Estes dados são utilizados para comparar a execução atual com as anteriores.

O histórico principal armazena as informações que impactam ou descrevem o tempo de execução de uma tarefa, são elas: A posição inicial (**Origem**); A posição objetivo (**Objetivo**); O tempo necessário

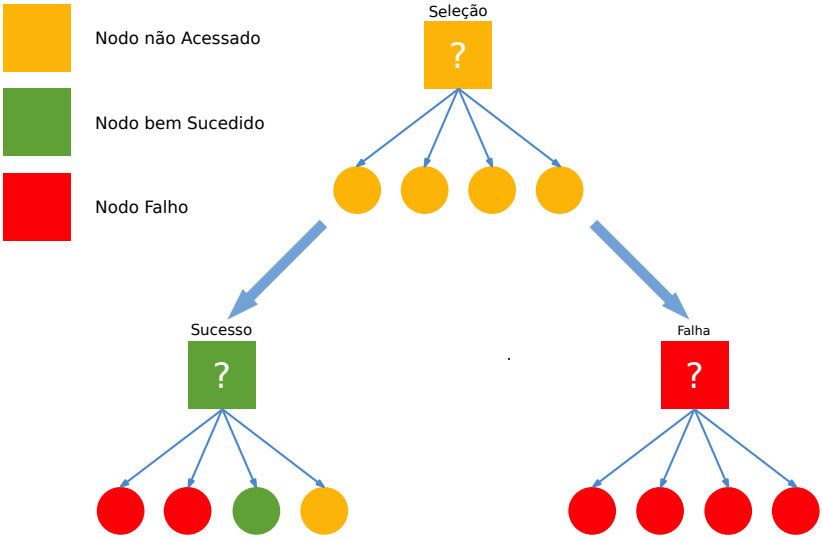


Figura 2.3: As condições de sucesso (à esquerda) e falha (à direita) para nodos de seleção (adaptado de [SIQUEIRA; PIERI, 2015])

para que o robô chegue ao ponto objetivo (**TParcial**); Tempo necessário para que o robô volte ao ponto inicial a partir do ponto objetivo (**TRetorno**); Número de obstáculos conhecidos (**NO**); Posição dos obstáculos (**Posições**) e; Velocidade média durante a navegação (**VM**). Sempre que uma execução da tarefa é completada, o histórico principal é atualizado com as informações coletadas. Este histórico possui um tamanho máximo e, ao atingir esse limite, as informações mais antigas são substituídas pelas mais recentes. Antes de iniciar a execução de uma tarefa, o robô recupera as informações do histórico antes de começar a se mover. A Tabela 2.1 apresenta o exemplo de um trecho do histórico principal.

2.2.2. Equação de Previsão de Monteiro

O mecanismo atua em duas formas: modo de aprendizagem e modo de previsão. O primeiro serve para preencher o histórico com informações sobre o ambiente para que estas possam ser usadas na previsão de perda de *deadline*. O segundo modo é o principal no qual o robô executa as tarefas e consulta o histórico para tentar prever se conseguirá realizar a tarefa dentro do prazo especificado ou não.

Origem	Objetivo	TP	TT	NO	Posições	VM
(123,664)	(987, 1245)	47.1	106.2	5	M1=(873, 232)...	900
(435,344)	(667, -7463)	52.8	120.2	6	C3=(987, -456)...	900
(123,664)	(987, 1245)	46.8	105.6	5	M1=(858, 238)...	900
(435,344)	(667, -7463)	53.4	121.5	6	C3=(972, -474)...	900
(123,664)	(987, 1245)	58.8	124.2	5	M1=(865, 232)...	500
...	...	...	...	...	...	...

Tabela 2.1: Exemplo de Trecho do Histórico Principal [MONTEIRO, 2014]

Antes de iniciar uma tarefa, o robô lê o histórico para obter informações sobre a execução da tarefa. A previsão de perda de *deadline* é calculada, usando informações da navegação atual e dados do histórico, quando o robô chega no ponto objetivo. Dado um certo *deadline* DL e o tempo parcial atual TPA (tempo necessário para o robô navegar do ponto inicial ao ponto objetivo), o tempo restante disponível TR (Tempo Restante) para completar a tarefa dentro do prazo estipulado é obtido pela Equação 2.1.

$$TR = DL - TPA \tag{2.1}$$

Tendo o tempo restante TR para completar a tarefa sem perder o *deadline*, o mecanismo de previsão deve calcular o tempo estimado de retorno (TER). Este valor é calculado ao analisar a média do tempo parcial de execução (TMP), a média do tempo total de execução (TMF), o tempo necessário para desviar de obstáculos fixos desconhecidos ($TDOF$) e uma folga de tempo reservada para desviar de obstáculos móveis encontrados na ida que ainda podem estar presentes na volta ($FDOM$) que podem aparecer durante a execução da tarefa. A Equação 2.2 mostra como o tempo estimado de retorno é calculado. A previsão é calculada pela função $f(DL)$, mostrada na equação 2.3.

Se TR for igual ou maior que o tempo estimado para que o robô retorne à posição inicial (TER), o mecanismo conclui que, provavelmente, o robô será capaz de completar a tarefa dentro do intervalo de tempo definido para aquela tarefa, ou seja, não perderá o *deadline*.

Caso TR seja menor que TER, então o robô provavelmente perderá o *deadline* e ações corretivas podem ser tomadas pela aplicação para evitar ou tentar minimizar possíveis perdas.

$$TER = TMF - TMP + TDOF + FDOM \tag{2.2}$$

$$f(DL) = \begin{cases} 1 & \text{se TR} \geq \text{TER} \\ 0 & \text{caso contrário.} \end{cases} \tag{2.3}$$

O mecanismo de previsão presume que a mesma quantidade de obstáculos encontrada na ida será, também, encontrada na volta, portanto, o tempo estimado de retorno, que deveria ser similar ao resultado da subtração entre o tempo médio final (TMF) e o tempo médio parcial (TMP), deve ser acrescido de alguns valores, indicando os atrasos que possam vir a ocorrer. Estes atrasos podem ser causados por algum objeto desconhecido encontrado no caminho ao objetivo e que precisou ser desviado. O mecanismo de previsão calcula estes atrasos por meio dos valores $TDOF$ e $TDOM$ na equação 2.2.

O valor de $TDOF$ será zero caso nenhum obstáculo estático e desconhecido tenha sido encontrado no caminho entre o ponto inicial e o ponto objetivo. Se o robô tiver encontrado obstáculos estáticos e desconhecidos, então o valor de $TDOF$ será igual ao resultado obtido pela subtração entre o tempo parcial atual (TPA), o tempo médio parcial (TMP) e o tempo de desvio de obstáculos móveis ($TDOM$), como mostrado na Equação 2.4.

$$TDOF = TPA - TMP - TDOM \tag{2.4}$$

O Valor de $TDOM$ é calculado pelo componente de detecção de obstáculos. Quando um obstáculo móvel é detectado ao longo do caminho do robô, este reduz sua velocidade ou pode até parar completamente, em situações mais extremas, até que obstáculo saia do caminho. O componente de detecção de obstáculos ativa um *timer* sempre que um obstáculo móvel é detectado no caminho do robô e o força a reduzir a velocidade ou parar. O *timer* é parado quando o obstáculo deixa o caminho do robô, o qual fica livre para navegar. O valor final do *timer* quando o robô chega ao objetivo é o valor de $TDOM$.

Durante o trajeto do ponto inicial ao ponto objetivo, o robô pode encontrar obstáculos móveis. Devido a sua natureza móvel, estes obstáculos podem não se encontrar no caminho durante o retorno ao ponto inicial, entretanto isto não será sempre verdade. Para considerar a possibilidade de estes obstáculos ainda se encontrarem no caminho

durante a volta, um valor referente a uma folga de desvio de obstáculos móveis ($FDOM$) é adicionado ao cálculo do tempo estimado de volta (TER). $FDOM$ é resultado da multiplicação de uma constante $Folga$, definida pelo usuário, com o número de obstáculos móveis encontrados (NOM) durante a ida do ponto inicial ao ponto objetivo. A equação 2.5 mostra o cálculo previamente citado.

$$FDOM = NOM \times Folga \tag{2.5}$$

Há a possibilidade de o robô não encontrar quaisquer obstáculos durante a navegação do ponto inicial ao ponto objetivo, caso isto o ocorra, os valores de $TDOF$ e $FDOM$ serão nulos e a equação que calcula o tempo estimado de retorno (TER) pode ser simplificada, conforme mostra a Equação 2.6.

$$TER = TMF - TMP \tag{2.6}$$

O mecanismo de previsão proposto por Monteiro [2014] possui uma taxa de acerto de, aproximadamente, 90%, o que o torna um mecanismo bastante confiável. Esta taxa pode cair caso haja um aumento no número de obstáculos desconhecidos após a previsão ter sido realizada. Monteiro trata desta questão de forma apropriada em seu trabalho.

2.3. Previsão de Perda de *Deadline* para Grupos de Robôs Móveis

Este trabalho modificou o mecanismo proposto por [MONTEIRO, 2014] para um grupo de robôs. O contexto de execução é o mesmo daquele trabalho, ou seja, ambientes total ou parcialmente conhecidos e o modelo de tarefas em que o mecanismo é aplicado consiste na navegação de um ponto inicial a um ponto objetivo e no retorno ao ponto inicial.

Como múltiplos robôs são utilizados neste trabalho, o histórico é associado a uma tarefa, ao invés de ser associado a um robô, o que permite que qualquer tarefa possa ser realizada por qualquer robô do grupo. O histórico principal proposto por Monteiro foi redefinido para armazenar as seguintes informações: a posição inicial; a posição objetivo; o tempo necessário para que o robô chegue ao ponto objetivo; o tempo necessário para que o robô volte ao ponto inicial a partir do ponto objetivo; o tempo total de execução; o resultado da previsão; uma variável indicando se o robô foi capaz de cumprir o *deadline*; o

número de obstáculos encontrados; o número de objetos carregados; o *deadline* e; a velocidade média durante a navegação.

Algumas das equações utilizadas no processo de previsão foram modificadas e são apresentadas na subseção a seguir.

2.3.1. Cálculo da previsão

Dado um certo *deadline* DL e o tempo parcial atual TPA (tempo necessário para o robô navegar do ponto inicial ao ponto objetivo), o tempo restante disponível TR (Tempo Restante) para completar a tarefa dentro do prazo estipulado é obtido pela Equação 2.1.

Neste trabalho, o TER é calculado ao se analisar a média do tempo de retorno de execuções anteriores daquela tarefa ($\overline{X_{TR}}$), o tempo necessário para desviar de obstáculos fixos desconhecidos ($TDOF$) e uma folga de tempo reservada para desviar de obstáculos móveis encontrados na ida que ainda podem estar presentes na volta ($FDOM$) que podem aparecer durante a execução da tarefa. O cálculo do tempo estimado de retorno considera um intervalo de confiança (IC) de 95%, obtido do histórico, conforme mostra a Equação 2.7. A previsão é calculada, assim como no trabalho de Monteiro pela função $f(DL)$, mostrada na equação 2.3.

$$TER = \overline{X_{TR}} + \beta \times IC + \text{TDOF} + \text{FDOM} \tag{2.7}$$

O algoritmo pode ter uma abordagem otimista, pessimista ou neutra, mostrada na Equação 2.8. Para a abordagem otimista, o algoritmo usa o limite inferior do intervalo de confiança, significando que o tempo de retorno será, provavelmente, menor do que a média. Em uma abordagem pessimista, o algoritmo usa o limite superior, o que significa que o tempo de retorno provavelmente será maior que a média. A abordagem neutra usa o tempo médio de retorno, o que significa que o tempo de retorno será, provavelmente, o mesmo que a média.

$$\beta = \begin{cases} 1 & \text{se pessimista} \\ 0 & \text{se neutro} \\ -1 & \text{se otimista} \end{cases} \tag{2.8}$$

$TDOF$ (Tempo de Desvio de Obstáculos Fixos) assume valor igual a zero caso nenhum obstáculo fixo seja encontrado (mesma abordagem utilizada em [MONTEIRO, 2014]). Caso contrário, seu valor é

definido conforme a Equação 2.9.

$$TDOF = \begin{cases} 0 & \text{se TPA} \in [TMP - IC; TMP + IC] \\ \text{TPA - TMP} & \text{caso contrário.} \end{cases} \quad (2.9)$$

De acordo com esta equação, $TDOF$ será zero se o tempo parcial atual (TPA) se encontra dentro do intervalo de confiança (IC) do tempo médio parcial (TMP) com 95% de confiança. Isto significa que, se o valor resultante da diferença entre o tempo parcial de execução e o tempo necessário para desviar de obstáculos móveis for similar ao tempo médio parcial, obtido a partir do histórico, com 95% de confiança, então o obstáculo não aumentou o tempo de execução da tarefa e, logo, $TDOF$ será zero. Por outro lado, se TPA não está no intervalo de confiança de TMP, então o valor de $TDOF$ será igual ao resultado da subtração entre o tempo parcial atual (TPA) e o tempo médio parcial (TMP).

Caso o robô não encontre nenhum obstáculo durante a navegação do ponto inicial ao ponto objetivo, a equação que calcula o tempo estimado de retorno (TER) é simplificada, conforme mostra a Equação 2.10.

$$TER = \overline{X_{TR}} + \beta IC \quad (2.10)$$

2.4. Proposta de Ações Corretivas para Perdas de *Deadline*

Este trabalho propõe o desenvolvimento de ações corretivas para serem executadas quando um robô móvel prevê uma perda de *deadline* da tarefa que está executando. Estas ações corretivas têm como objetivo auxiliar o robô a terminar a tarefa dentro das restrições temporais especificadas para ela ou reduzir os prejuízos que podem ser causados pelo não cumprimento destas restrições temporais.

Cada tarefa possui um histórico associado a ela. Neste histórico são armazenadas informações sobre as navegações realizadas, conforme explicado na seção 2.3.

A Figura 2.4 apresenta um exemplo de uso desse mecanismo no caso de algum dos robôs do grupo detectar uma perda de *deadline*. Nas figuras, os círculos cinza representam os robôs, as linhas em cada círculo indicam a orientação do robô e as linhas tracejadas representam as trajetórias a serem seguidas por cada robô. Na Figura 2.4a

os robôs devem se deslocar do ponto em que estão (ponto inicial) ao ponto objetivo (círculos vermelhos), o robô "*robot_3*" está em espera em uma doca de carregamento. Na Figura 2.4b, os robôs chegam ao ponto objetivo e realizam os cálculos de previsão para ver se conseguirão retornar ao ponto inicial dentro da restrição temporal especificada. Nesta ocasião, o robô "*robot_1*" acusa perda de *deadline*. Para reduzir o prejuízo que pode ser causado pelo não cumprimento da restrição temporal, o mecanismo transfere os dados da tarefa realizada por "*robot_1*" para "*robot_3*" e aquele recebe o comando para se dirigir à doca. Na Figura 2.4c, "*robot_3*" se desloca ao ponto inicial da tarefa que antes era realizada por "*robot_1*", enquanto este se desloca para a doca de carregamento (trajetória demonstrada pela linha pontilhada). Os outros dois robôs continuam realizando suas tarefas normalmente. Na Figura 2.4d, "*robot_3*" já está executando a tarefa e "*robot_1*" está recarregando, enquanto espera que algum outro robô do grupo acuse a perda de *deadline* ou apresente um nível de bateria abaixo dos valores aceitáveis para a execução da tarefa.

2.5. Conclusão

Este capítulo apresentou conceitos relevantes para o desenvolvimento deste trabalho. Na seção 2.3, são apresentadas modificações realizadas em algumas das equações do mecanismo proposto por Monteiro [2014]. A proposta de pesquisa também foi apresentada neste capítulo. O próximo capítulo apresenta as ferramentas utilizadas para o desenvolvimento do sistema e uma descrição deste e de seus módulos. A forma como as simulações foram executadas, bem como os cenários nos quais elas ocorreram e os resultados obtidos em cada cenário são, também, apresentados no próximo capítulo.

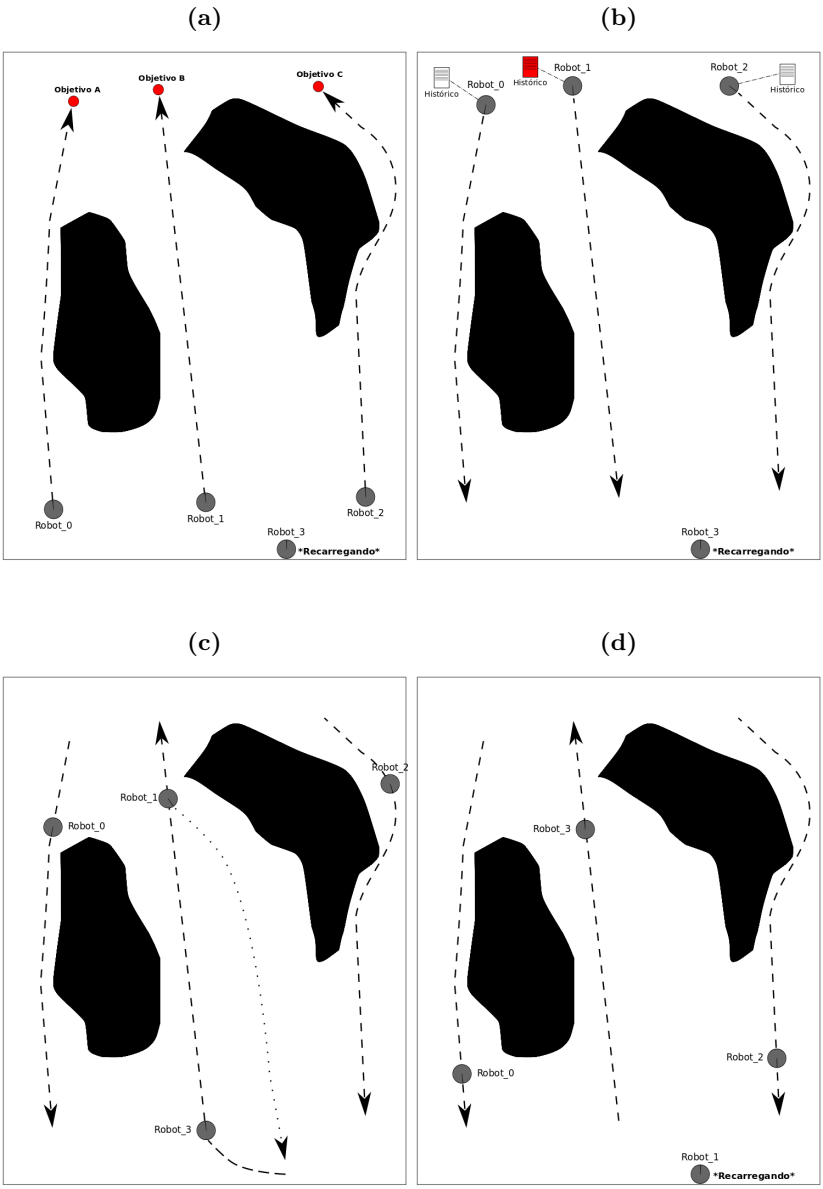


Figura 2.4: Exemplo de tarefa realizada pelos robôs móveis e situação em que a perda de *deadline* é prevista por um dos robôs do grupo

Capítulo 3

Implementação, Simulações e Resultados

Neste capítulo são apresentadas as ferramentas utilizadas para o desenvolvimento do sistema, uma descrição sobre seus módulos, a forma como as simulações foram executadas, os cenários nos quais elas foram realizadas e os resultados obtidos em cada um deles. A seção 3.1 apresenta o meta-sistema operacional no qual o mecanismo implementado executa, bem como as bibliotecas e a linguagem de programação utilizadas em sua implementação. Na seção 3.2, o sistema é descrito e seus módulos são explicados. A seção 3.3 discorre, de forma geral, sobre como as simulações foram executadas. Os cenários de simulação são apresentados na seção 3.4. Na seção 3.6, os resultados obtidos em cada cenário são apresentados. Uma conclusão referente aos resultados alcançados é apresentada na seção 3.7.

3.1. Implementação

O trabalho proposto neste documento foi implementado utilizando a distribuição *Indigo* do *Robot Operating System* (*ROS*), que se trata de um meta-sistema operacional de código aberto que fornece diversos serviços, tais como abstração de hardware, controle de dispositivos de baixo nível, troca de mensagens entre processos e gerenciamento de pacotes [O'KANE, 2013].

Em seguida, são listadas algumas das vantagens em utilizar o ROS:

- Computação Distribuída: Robôs que realizam uma mesma tarefa precisam se comunicar, além disso um operador humano pode controlar um robô remotamente por meio de um computador, *laptop* ou dispositivo móvel. Mesmo em um único computador, é possível dividir o software do robô em pequenas partes

autônomas que cooperam entre si. O ROS fornece suporte às características citadas por meio de seu mecanismo de comunicação;

- Reutilização de Software: Os pacotes padrões do ROS fornecem versões estáveis de diversos algoritmos importantes para a robótica (navegação e planejamento de trajetória, por exemplo);
- Testes Rápidos: Sistemas bem projetados, que utilizam o ROS, separam o controle direto de baixo nível do software de alto nível, permitindo que este seja testado em simuladores para verificar seu comportamento. O ROS fornece, também, uma forma de gravar e executar dados de sensores, o que permite que esses dados gravados sejam reexecutados várias vezes para testar diferentes formas de processá-los.

Além destas vantagens, o ROS também possui uma extensa documentação *online*[1], onde é possível encontrar tutoriais, códigos e informações sobre as versões disponíveis. Há também uma página de suporte da comunidade[2].

Um dos objetivos do ROS é permitir que o código desenvolvido seja uma coleção de programas pequenos e praticamente independentes chamados "nós". Todos os nós executam simultaneamente e, para que isso funcione, eles precisam se comunicar. O *ROS Master* é o nó responsável por facilitar essa comunicação. Este nó deve estar em execução durante todo o tempo em que o ROS estiver sendo utilizado, dado que ele é o responsável por realizar a conexão entre os nós que estiverem em execução [O'KANE, 2013].

O mecanismo de comunicação utilizado pelo ROS é a troca de mensagens. Um nó que deseja compartilhar informações irá publicá-las em um tópico específico para o tipo de mensagem a ser publicada, enquanto que um nó que queira receber informações irá se inscrever em um tópico que possua os dados buscados por ele. O *ROS Master* é responsável por permitir que os "publicadores" e os "inscritos" possam localizar um ao outro. Uma vez que esses nós consigam se encontrar, eles iniciam uma comunicação ponto-a-ponto. A Figura 3.1 possui um grafo representando a estrutura de comunicação entre diversos nós. Nesta figura, as setas indicam o sentido das mensagens, mostrando quais nós publicam e quais recebem as mensagens de cada tópico.

O ROS possui suporte total a *C++*, *Python* e *Lisp* e possui algumas bibliotecas experimentais implementadas em *Java* e *Lua*. Dentre

[1] <http://wiki.ros.org>
[2] <http://answers.ros.org>

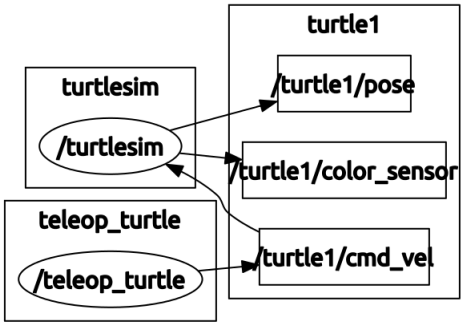


Figura 3.1: Grafo apresentando a comunicação entre os nós.

as linguagens de programação citadas, escolheu-se a linguagem *C++* para implementar o mecanismo de previsão e as ações corretivas.

O simulador utilizado é o *Stage*[3]. A partir de um arquivo "*.world*", o *Stage* consegue simular um mapa para a execução dos testes. Todas as características do mapa estão descritas neste arquivo, desde a posição dos obstáculos ao tamanho dos robôs e informações sobre seus sensores, como o laser ou a câmera, por exemplo. Este simulador foi escolhido por já apresentar uma integração com o ROS. A Figura 3.2 mostra uma captura de tela do simulador *Stage*. o quadrado azul na figura representa o robô, a área em verde é o alcance do *laser* e as linhas pretas são a delimitação do ambiente. No canto inferior esquerdo é possível observar o tempo decorrido de simulação.

Para o planejamento de trajetória, foi utilizado o *move_base*[4], um nodo do *ROS* que realiza tanto o planejamento de trajetória quanto a própria navegação. Para a implementação da Árvore de Comportamento, foi utilizada uma biblioteca do ROS chamada *behavior_tree*, disponível em um repositório do *GitHub*[5].

3.2. Sistema

A Figura 3.3 apresenta a estrutura geral do sistema proposto. Nesta figura, as setas indicam o sentido das mensagens, os retângulos

[3]<http://wiki.ros.org/stage_ros>
[4]<http://wiki.ros.org/move_base>
[5]<https://github.com/almc/behavior_trees>

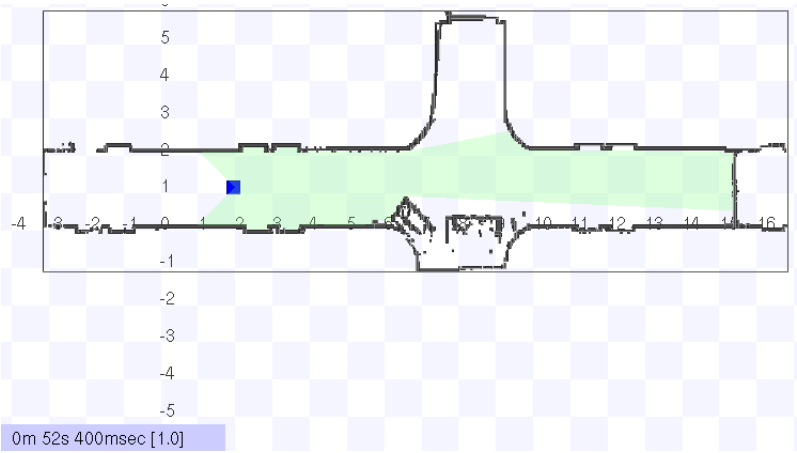


Figura 3.2: Captura de tela do simulador *Stage* executando um mapa carregado a partir de um arquivo "*.world*".

maiores indicam os módulos do sistema e os menores representam as funções realizadas por cada módulo. Os retângulos azuis retratam módulos e funções já implementados em bibliotecas do ROS, enquanto os laranjados indicam módulos e funções implementadas neste trabalho.

O Algoritmo 1 apresenta um pseudocódigo para o módulo *Linker*. Este módulo tem a função de distribuir as tarefas entre os robôs móveis disponíveis e também é responsável por manipular as informações referentes ao estado das tarefas sendo realizadas (como a informação de que algum robô previu a perda do *deadline* da tarefa que está realizando, por exemplo). Estas informações podem ser utilizadas em algumas das ações corretivas presentes na Árvore de Comportamento.

O módulo que representa a Árvore de Comportamento, na Figura 3.3 é, na verdade, uma subestrutura que compreende todos os nodos que constituem esta árvore. Este módulo possui as seguintes funções:

- Iniciar a navegação: A posição dos objetivos, utilizada pelo nodo *move_base* para gerar a trajetória, é enviada por um nodo de ação pertencente à Árvore de Comportamento;
- Mecanismo de previsão de perda de *deadline*: A Árvore de Comportamento tem o papel de prever se o robô será capaz de respeitar as restrições temporais da tarefa que está realizando;
- Armazenamento de informações da tarefa: Os dados obtidos durante a execução da tarefa são armazenados em um histórico para auxiliar no processo de previsão de perda de *deadline*;

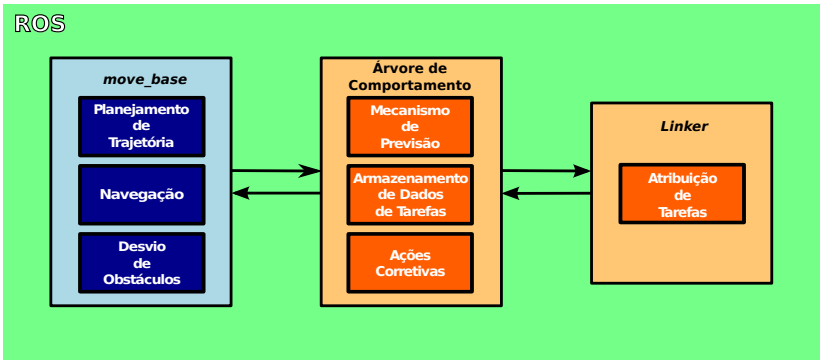


Figura 3.3: Estrutura do Sistema

- Ações corretivas: A Árvore de Comportamento possui um conjunto de ações corretivas para serem realizadas com o intuito de auxiliar o robô a cumprir uma tarefa respeitando suas restrições temporais ou para reduzir o prejuízo causado pelo não cumprimento do *deadline* desta tarefa.

3.2.1. Comportamento do Sistema

As diferentes ações que podem ser executadas foram divididas em três diferentes subárvores: a subárvore de ***Verificação de Ociosidade***, a subárvore ***Conjunto Normal de Ações*** e a subárvore ***Conjunto de Ações Corretivas***. A primeira possui os nodos utilizados para verificar se o robô está ocioso e, se esta condição for verdadeira, esperar até que algum robô ativo notifique uma possível perda de *deadline*. Na segunda subárvore estão as ações realizadas durante a execução normal de uma tarefa, o que significa que os nodos responsáveis pela navegação, pelo armazenamento de dados no histórico e pela previsão de perda de *deadline* estão nesta subárvore. A última subárvore possui ações que podem recuperar uma possível perda de *deadline* ou reduzir o prejuízo causado pelo não cumprimento das restrições temporais de uma tarefa. A estrutura principal da Árvore de Comportamento é apresentada na Figura 3.4.

A subárvore de *Verificação de Ociosidade* possui dois nodos condicionais, o nodo ***Está Ocioso?*** e o nodo ***Perda de Deadline não Detectada***. A estrutura desta subárvore está representada na Figura 3.5 e seus nodos são:

- ***Está Ocioso?***: Por meio de uma variável de controle, este nodo

Algoritmo 1 Linker.c

```
1: definição das posições inicial e final da tarefa
2: se nenhum robô foi atribuido às tarefas então
3:     definição do valor do tempo total de simulação;
4:     atribuição das tarefas aos robôs;
5:     Atribui o nome do robô ocioso à variável robô_ocioso;
6: senão se (perda_de_deadline_detectada == verdadeiro) e
   (perda_de_deadline_verificada == verdadeiro) então
7:     Verifica qual tarefa o robô que previu perda de deadline está executando;
8:     Atribui a tarefa ao robô ocioso;
9:     robô_ocioso ← Nulo;
10:    perda_de_deadline_verificada ← 0;
11: fim se
12: se robô chegou à doca de carregamento então
13:    robô_ocioso ← previu_perda;
14:    previu_perda ← Nulo;
15:    perda_de_deadline_detectada ← 0;
16: fim se
```

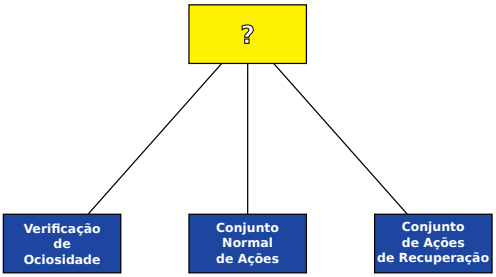


Figura 3.4: Estrutura principal da Árvore de Comportamento

realiza uma verificação para identificar se o robô que o executa está ocioso ou ativo. A variável de controle informa a todos os robôs qual deles está ocioso no momento. A cada execução deste nodo, cada robô verifica se ele é o robô indicado pela variável, caso seja, um *status* de sucesso é retornado. Caso o robô não esteja ocioso, este nodo retornará um *status* de falha e o robô iniciará a realizar a tarefa normalmente. O pseudocódigo representando o funcionamento deste nodo é apresentado no Algoritmo 2;

- ***Perda de Deadline não Detectada***: Este nodo verifica se al-

gum robô ativo notificou uma possível perda de *deadline*. Caso o não cumprimento das restrições temporais da tarefa seja detectado, este nodo retornará falha e, caso nenhum robô do grupo tenha previsto perda de *deadline*, o nodo retornará sucesso. O Algoritmo 3 apresenta o pseudocódigo referente a este nodo.

Algoritmo 2 está_ocioso.c

```
1: se nome_do_robô == robô_ocioso então
2:     retorna Sucesso;
3: senão
4:     retorna Falha;
5: fim se
```

Algoritmo 3 perda_de_deadline_não_detectada.c

```
1: se perda_de_deadline_detectada então
2:     perda_de_deadline_verificada ← 1;
3:     retorna Falha;
4: senão
5:     retorna Sucesso;
6: fim se
```

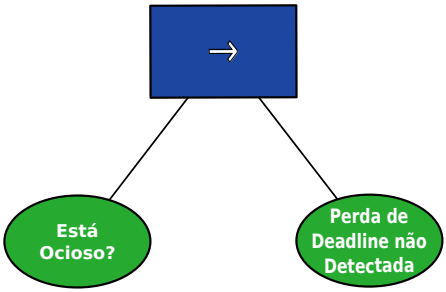


Figura 3.5: Subárvore de *Verificação de Ociosidade*

A subárvore *Conjunto Normal de Ações* compreende as ações realizadas durante uma execução normal da tarefa. Esta subárvore possui cinco nodos (três nodos de ação e dois condicionais) e sua estrutura está representada na Figura 3.6. Os nodos presentes nesta subárvore são:

- ***Histórico Suficiente***: Este nodo verifica se o histórico possui entradas suficientes para executar a tarefa com o mecanismo de previsão de perda de *deadline* funcionando corretamente. Este

nodo sempre retorna um *status* de sucesso, entretanto, a quantidade de dados no histórico determinará o valor de algumas variáveis de controle. Se não houver dados suficientes no histórico, o robô executará a tarefa no modo de aprendizagem, cuja função é preencher o histórico. O Algoritmo 4 mostra o comportamento deste nodo;

- ***Mover para Objetivo***: Quando este nodo está ativo, o robô navega até a posição objetivo da tarefa. Durante a navegação, algumas informações, como o tempo necessário para se alcançar o objetivo e a quantidade de objetos encontrados no caminho, são mantidas em variáveis para serem utilizadas no mecanismo de previsão e para serem armazenadas no histórico. Este nodo retorna um *status* de sucesso caso o robô alcance o ponto objetivo e retorna falha caso o robô não possa alcançá-lo. O pseudocódigo apresentado no Algoritmo 5 mostra o funcionamento deste nodo;
- ***Previsão de Cumprimento de Deadline***: Neste nodo, a previsão de perda de *deadline* é realizada, conforme apresentado no Algoritmo 6, que apresenta o pseudocódigo referente a este nodo. Caso um *status* de falha seja retornado por este nodo, a execução normal da tarefa será interrompida e o robô executará alguma das ações presentes na subárvore *Conjunto de Ações Corretivas*. Este nodo retorna um *status* de sucesso se o mecanismo confirma que as restrições temporais da tarefa atual serão respeitadas ou se o robô está executando esta tarefa no modo de aprendizagem;
- ***Mover para Início***: Neste nodo, o robô navega de volta à posição inicial da tarefa. Como no nodo *Mover para Objetivo*, algumas informações são armazenadas em variáveis para, posteriormente, serem adicionadas ao histórico. Ao fim da navegação, o robô verifica se foi capaz de cumprir a tarefa sem perder o *deadline*. Este nodo retorna um *status* de sucesso ou falha seguindo as mesmas regras do nodo *Mover para Objetivo*. O Algoritmo 7 traz o pseudocódigo que apresenta o funcionamento deste nodo;
- ***Armazenar Dados***: Os dados coletados durante toda a execução da tarefa são armazenados no arquivo de histórico durante a execução deste nodo. Como mostrado no pseudocódigo apresentado no Algoritmo 8, este nodo sempre retorna um *status* de sucesso.

A subárvore *Conjunto de Ações Corretivas* contém as ações a serem realizadas quando o robô prevê uma perda de *deadline* durante a execução de uma tarefa. Os nodos desta subárvore têm o propósito de fazer com que o robô seja capaz de respeitar as restrições temporais da

Algoritmo 4 histórico_suficiente.c

```
1: se deadline não definido então
2:     Recupera dados do histórico;
3: fim se
4: se histórico tem mínimo de entradas então
5:     tempo_médio_de_retorno ← histórico.tempo_médio_de_retorno;
6:     tempo_médio_parcial ← histórico.tempo_médio_parcial;
7:     tempo_médio_total ← histórico.tempo_médio_total;
8:     intervalo_de_confiança ← histórico.obter_intervalo_de_confiança;
9:     deadline ← tempo_médio_total;
10:    se momento de início da simulação não definido então
11:        início_da_simulação ← tempo_atual();
12:    fim se
13: fim se
14: retorna Sucesso;
```

Algoritmo 5 mover_para_objetivo.c

```
1: enquanto move_base não responder faça
2:     objetivo_ativo ← falso;
3: fim enquanto
4: se !objetivo_ativo então
5:     envia objetivo ao move_base;
6:     objetivo_ativo ← verdadeiro;
7: fim se
8: se status_da_navegação == terminada então
9:     retorna Sucesso;
10: senão se status_da_navegação == falha então
11:    retorna Falha;
12: senão
13:    retorna Em_Execução;
14: fim se
```

Algoritmo 6 previsão_de_cumprimento_de_deadline.c

```
previsão ← previsão_de_cumprimento_de_deadline();
se previsão então
    se tempo_médio_de_retorno > tempo_total_de_simulação - (tempo_atual() - início_da_simulação) então
        retorna Falha;
    senão
        número_de_objetos ← 2;
        peso_de_navegação ← 0.5;
    fim se
    retorna Sucesso;
senão se (!previsão) e (!robô_ocioso) e (!previu_perda) então
    retorna Falha;
senão
    se tempo_médio_de_retorno > tempo_total_de_simulação - (tempo_atual() - início_da_simulação) então
        retorna Falha;
    senão
        número_de_objetos ← 2;
        peso_de_navegação ← 0.5;
    fim se
    retorna Sucesso;
fim se
```

Algoritmo 7 mover_para_inicio.c

```
enquanto move_base não responder faça
    objetivo_ativo ← falso;
fim enquanto
se !objetivo_ativo então
    envia objetivo ao move_base;
    objetivo_ativo ← verdadeiro;
fim se
se status_da_navegação == terminada então
    verifica se o robô acertou a previsão;
    retorna Sucesso;
senão se status_da_navegação == falha então
    retorna Falha;
senão
    retorna Em_Execução;
fim se
```

Algoritmo 8 armazenar_dados.c

```
1: se (tempo_atual() - início_da_simulação) > tempo_total_de_simulação então
2:     retorna Sucesso;
3: fim se
4: total_objetos_carregados ← total_objetos_carregados + número_de_objetos;
5: armazena_dados();
6: retorna Sucesso;
```

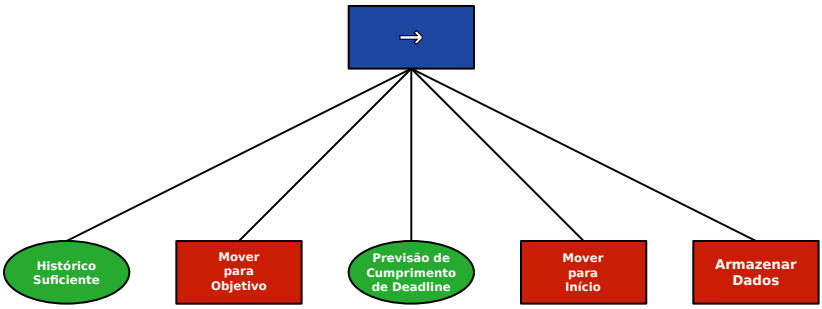


Figura 3.6: Subárvore *Conjunto Normal de Ações*

tarefa que está executando ou tentar reduzir o prejuízo causado pela perda do *deadline*. Esta subárvore tem seis nodos, sendo um de seleção e cinco nodos de ação, que são: ***Substituição de Robôs***, ***Tempo Insuficiente***, ***Mover para Início***, ***Armazenar Dados*** e ***Em Espera***. O nodo seletor tem como objetivo definir qual das ações corretivas será utilizada, de acordo com o estado atual da tarefa. A estrutura da subárvore *Conjunto de Ações Corretivas* é apresentada na Figura 3.7.

- ***Substituição de Robôs***: Caso uma perda de *deadline* tenha sido prevista e o tempo restante para que o robô realize as tarefas (*deadline* global) seja maior que a média do tempo de retorno, obtido do histórico, então esta ação corretiva será utilizada. Este nodo reduz a carga do robô que está realizando a tarefa e "avisa" ao robô ocioso que uma perda de *deadline* foi detectada. O *status* de falha é retornado caso o tempo restante para que o robô realize as tarefas (*deadline* global) seja menor que o tempo médio de retorno da tarefa, obtido do histórico. Será retornado sucesso em qualquer outra situação. O Algoritmo 9 apresenta o pseudocódigo referente a este nodo.

- ***Tempo Insuficiente***: Este nodo é executado no caso do tempo restante para a realização de tarefas (*deadline* global) ser menor que o tempo médio de retorno, obtido a partir dos dados armazenados no histórico. Durante a execução deste nodo, a quantidade de objetos carregados pelo robô que está executando a tarefa é reduzida e, ao fim de sua execução, o *status* de sucesso é sempre retornado. O pseudocódigo representando o funcionamento deste nodo é apresentado no Algoritmo 10;
- ***Em Espera***: Este nodo, assim como o pseudocódigo apresentado no Algoritmo 11, enviará o robô para uma doca de carregamento disponível. Esta ação é realizada quando o robô regressa para o ponto inicial após ter realizado alguma ação corretiva. Este nodo retorna um *status* de sucesso se o robô pode chegar à doca de recarga e um *status* de falha caso não consiga.

Nesta subárvore, os nodos *Mover para Início* e *Armazenar Dados* realizam as mesmas funções que os nodos homônimos da subárvore *Conjunto Normal de Ações*.

Algoritmo 9 substituição_de_robôs.c

```
1: se tempo_médio_de_retorno > tempo_total_de_simulação -
   (tempo_atual() - início_da_simulação) então
2:     previu_perda ← nome_do_robô;
3:     número_de_objetos ← 1;
4:     peso_de_navegação ← 0.75;
5:     retorna Sucesso;
6: senão
7:     retorna Falha;
8: fim se
```

Algoritmo 10 tempo_insuficiente.c

```
1: número_de_objetos ← 1;
2: peso_de_navegação ← 0.75;
3: retorna Sucesso;
```

3.2.2. Cenário de Simulação

Para validar o sistema, foi proposto um cenário de simulação no qual um grupo de robôs atua. Considerou-se os cenários de um campus universitário e de um condomínio de apartamentos e os robôs deveriam

Algoritmo 11 em_espera.c

```
1: enquanto move_base não responder faça
2:     objetivo_ativo ← falso;
3: fim enquanto
4: se !objetivo_ativo então
5:     envia objetivo ao move_base;
6:     objetivo_ativo ← verdadeiro;
7: fim se
8: se status_da_navegação == terminada então
9:     avisa ao nodo Linker que chegou a doca de carregamento;
10:    retorna Sucesso;
11: senão se status_da_navegação == falha então
12:    retorna Falha;
13: senão
14:    retorna Em_Execução;
15: fim se
```

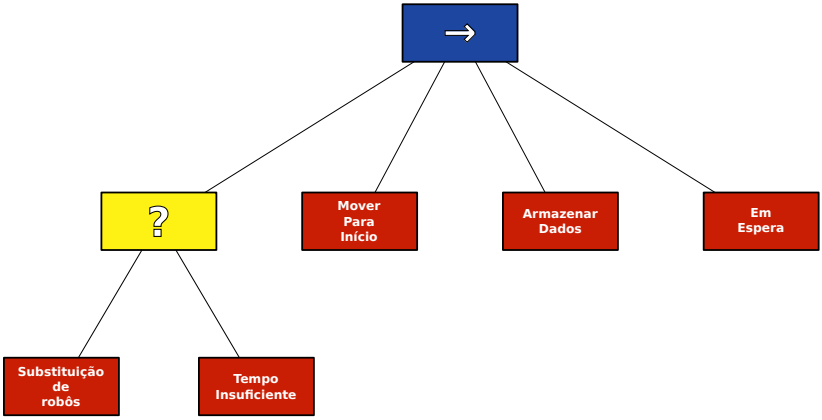


Figura 3.7: Subárvore *Conjunto de Ações Corretivas*

realizar tarefas de coletar itens de diferentes locais e descarregá-los em um local específico.

A localização dos itens é conhecida pelos robôs e cada local possui um conjunto de itens para ser carregado. Os robôs devem levar os itens a um ponto de entrega dentro de um intervalo de tempo.

Foi definido este tipo de tarefa para validar o sistema porque permite verificar o comportamento do mecanismo de previsão e as vantagens em utilizar as ações corretivas para tentar evitar ou minimizar as possíveis perdas causadas pelo não cumprimento de um *deadline*.

Um tipo de tarefa que possui a mesma configuração da definida como cenário de simulação é uma tarefa de coleta de lixo. Os robôs coletarão o lixo de diferentes localizações e os descarregarão em um local no qual, posteriormente, o lixo será coletado. Considerando que não haverá lixo para ser coletado o tempo todo, os robôs são ativados apenas em determinados momentos do dia, que são definidos em períodos nos quais há pouca movimentação de pessoas ou veículos no ambiente.

3.3. Simulações

As simulações foram realizadas em três diferentes mapas, que refletem três níveis de complexidade. Inicialmente, os robôs executam as tarefas, em cada um dos mapas, sem a presença de obstáculos móveis ou desconhecidos. Dessa forma, o histórico é preenchido com a quantidade mínima de entradas para que o mecanismo possa funcionar adequadamente. Em cada simulação, a tarefa realizada consiste no deslocamento do robô de um ponto inicial a um ponto objetivo e no retorno ao ponto de origem. A cada execução da tarefa o robô deve carregar consigo até dois objetos e cada objeto reduz a velocidade do robô em 25%. Durante a execução, os robôs devem prever se serão capazes de respeitar as restrições temporais de cada tarefa, caso não sejam, eles devem realizar alguma ação corretiva para tentar reduzir os prejuízos causados pela perda de *deadline*. Nas simulações realizadas, são levadas em consideração duas diferentes ações corretivas: reduzir a carga para aumentar a velocidade e anunciar a perda de *deadline* para o robô reserva, para que este assuma a tarefa; ou apenas reduzir a carga para aumentar a velocidade, caso o tempo médio de retorno para o ponto inicial seja superior ao tempo que resta do *deadline* global.

São adotados dois diferentes *deadlines* para as simulações: o *deadline* global e o *deadline* local. O primeiro se refere ao tempo máximo de simulação definido e o segundo é a média do tempo total que a tarefa leva para ser concluída. Este *deadline* é obtido a partir dos dados armazenados em histórico, enquanto aquele é definido pelo usuário. Como a tarefa simulada consiste no carregamento de objetos em um ambiente parcialmente conhecido, o usuário responsável pela definição do *deadline* global deve ser alguém com conhecimento do período de tempo em que haja uma menor movimentação de pessoas ou veículos no ambiente para que, dessa forma, o robô que realiza a tarefa tenha uma maior liberdade de movimento, ou seja, uma menor probabilidade de ter seu caminho obstruído por um obstáculo desconhecido. O *deadline* local é considerado pois ele indica aos robôs o tempo mé-

dio que uma tarefa leva para ser executada e, se for respeitado, pode garantir que o maior número possível de objetos seja carregado dentro do *deadline* global.

Quando um robô alcança a posição objetivo da tarefa, ele prevê se será capaz de respeitar as restrições temporais. Neste momento, uma das ações corretivas deverá ser executada caso o robô preveja uma perda de *deadline*. Na primeira ação corretiva citada anteriormente, o robô envia a informação de que possivelmente perderá o *deadline* para o robô ocioso, que está esperando em uma doca de carregamento. O robô que previu a perda de *deadline* irá pegar apenas um objeto do objetivo e irá levá-lo para a área de entrega. Carregar apenas um objeto permite que o robô se movimente mais rápido do que se estivesse carregando a maior quantidade de objetos possível, portanto o robô pode, mais uma vez, ser capaz de cumprir a tarefa respeitando seu *deadline*. Após deixar o objeto no ponto inicial, o robô se desloca para uma doca de carregamento disponível e espera até que algum outro robô do grupo preveja uma perda de *deadline*. Ao receber a informação de que algum robô previu uma perda de *deadline*, o robô que está atualmente ocioso irá trocar de lugar com aquele outro robô. Ele irá, primeiramente, ler o histórico da tarefa e coletar as informações que estão armazenadas e, então, começará a executar a tarefa normalmente, até que preveja uma perda de *deadline*.

A segunda ação corretiva citada é utilizada quando o *deadline* global é menor que a média do tempo de retorno. Neste caso, o robô reduz a quantidade de objetos que carrega, entretanto não avisa ao robô ocioso que não será capaz de cumprir as restrições temporais da tarefa. Isso se deve ao fato de o tempo para realizar as tarefas (*deadline* global) estar perto de terminar, o que tornaria inútil avisar outro robô, já que seria impossível que ele completasse a tarefa dentro do tempo restante.

Em cada ambiente de simulação há dois robôs móveis que atuam como obstáculos móveis. Estes robôs se movem em momentos aleatórios dentro de intervalos de tempo predefinidos.

Em cada um dos três cenários, dez simulações foram realizadas. Para cada cenário, o valor definido para o *deadline* global foi um período de tempo no qual o robô fosse capaz de executar tarefas o suficiente para que seu desempenho pudesse ser avaliado. Os intervalos de tempo no qual os robôs que atuavam como obstáculos se moviam foram determinados, para cada simulação, por meio de diversos experimentos. Com esses valores os robôs que atuam como obstáculos são capazes de obstruir o caminho dos robôs que estão executando a tarefa de modo

a influenciar no resultado de previsão, entretanto, apesar de os robôs obstáculos conseguirem obstruir o caminho em todas as simulações, a quantidade de vezes que eles bloqueiam os outros robôs é diferente a cada simulação.

3.4. Ambientes de Simulação

3.4.1. Mapa 1

Este mapa representa o corredor do 3º andar do INE e foi obtido a partir do mapa utilizado por Monteiro [2014]. As configurações iniciais deste mapa estão demonstradas na Figura 3.8. Nesta Figura, os círculos vermelho e azul indicam os pontos objetivo das tarefas 1 e 2 respectivamente, o retângulo amarelo representa o ponto de entrega dos objetos, a elipse verde indica a posição das docas de recarregamento, a elipse vermelha representa o ponto inicial dos robôs móveis utilizados como obstáculos e os círculos laranjados indicam sua posição objetivo.

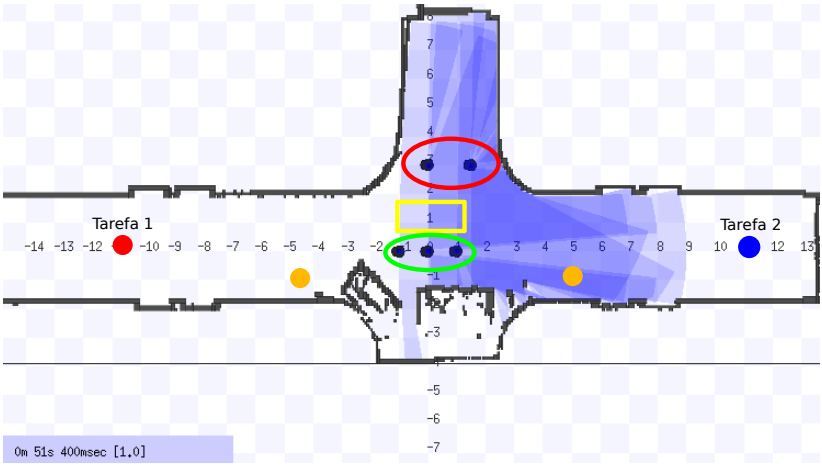


Figura 3.8: Configuração inicial Mapa 1

3.4.2. Mapa 2

Este mapa é a representação de uma parte do Centro Tecnológico (CTC) da Universidade Federal de Santa Catarina (UFSC). A imagem que serviu como base para a geração do mapa foi obtida através do

Google Maps[6]. A Figura 3.9 mostra a imagem de satélite utilizada como base para a geração do mapa e a Figura 3.10 mostra a configuração inicial para este mapa. Nesta Figura, os círculos vermelho e azul indicam os pontos objetivo das tarefas 1 e 2 respectivamente, os quadrados amarelos representam os pontos de entrega dos objetos, as elipses verdes indicam a localização inicial de cada robô, a posição das docas de recarregamento é representada pela elipse vermelha e os círculos laranjados indicam o ponto objetivo dos robôs móveis utilizados como obstáculos, sendo que cada ponto laranjado corresponde ao robô mais próximo.

Figura 3.9: Imagem de satélite de uma parte do Centro Tecnológico (CTC) da UFSC, obtida em 24 de Março de 2016

3.4.3. Mapa 3

Este mapa é a representação de um condomínio residencial. A imagem que serviu como base para a geração do mapa foi obtida através

[6] <https://www.google.com/maps/@-27.6001236,-48.5184769,96m/data=!3m1!1e3>

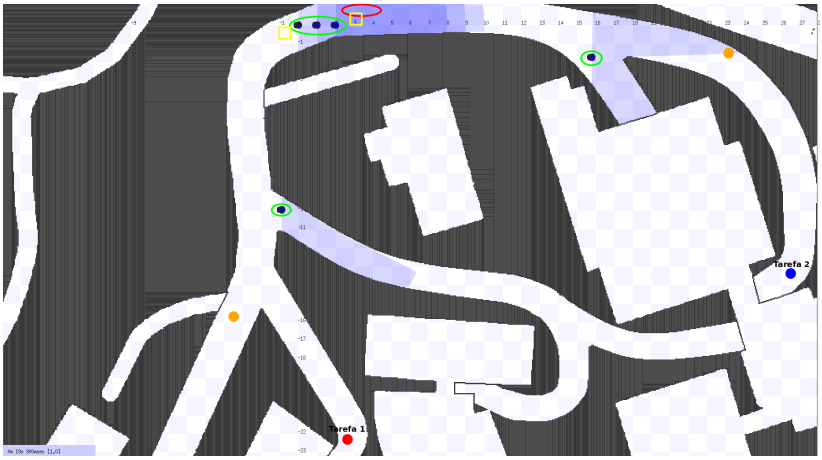


Figura 3.10: Configuração inicial Mapa 2

do Google Maps[7]. A Figura 3.11 mostra a imagem de satélite utilizada como base para a geração do mapa e a Figura 3.12 mostra a configuração inicial para este mapa. Nesta Figura, os círculos vermelho e azul indicam os pontos objetivo das tarefas 1 e 2 respectivamente, o retângulo amarelo representa o ponto de entrega dos objetos, as elipses verdes indicam a localização inicial de cada robô, a posição das docas de recarregamento é representada pela elipse vermelha e os círculos laranjados indicam o ponto objetivo dos robôs móveis utilizados como obstáculos, sendo que cada ponto laranjado corresponde ao robô mais próximo.

3.5. Tamanho Inicial de Histórico

Foram realizadas simulações com três tamanhos diferentes de histórico: 25, 50 e 100 entradas. Essas simulações foram necessárias para comprovar o impacto que o tamanho do histórico causa na definição do *deadline*. Para cada um dos três tamanhos iniciais de histórico foram realizadas dez simulações em cada um dos três mapas.

De acordo com os dados obtidos por meio das simulações, o tamanho do histórico inicial não causa muita influência na variação do valor do *deadline* local, que é calculado como a média do tempo total

[7] <https://www.google.com.br/maps/@-15.5694046,-56.0968103,111m/data=!3m1!1e3>

Figura 3.11: Imagem de satélite do condomínio residencial Miguel Sutil, obtida em 21 de Janeiro de 2016

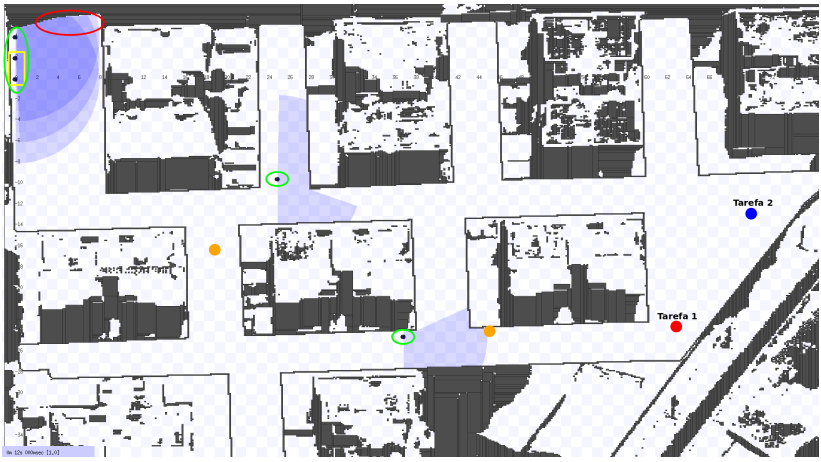


Figura 3.12: Configuração inicial Mapa 3

de execução da tarefa. Esta média é obtida a partir dos dados armazenados no histórico. Porém, apesar de não causar tanto impacto na

variação do *deadline*, o tamanho inicial do histórico influencia na estabilidade deste valor. Históricos maiores necessitam, geralmente, de mais simulações para que o valor do *deadline* de uma tarefa seja alterado. Isso se deve ao fato de que quanto maior for o tamanho do histórico, menor é a influência de uma nova entrada no resultado final do cálculo da média do tempo total, logo, para que haja uma alteração no valor do *deadline*, devem ser adicionados ao histórico uma grande quantidade de dados ou entradas com valores bastante discrepantes.

Na seção seguinte são apresentados os resultados para as simulações com tamanho inicial de histórico igual a 25. Os resultados para as simulações com histórico inicial igual a 50 e 100 se encontram nos apêndices A e B, respectivamente.

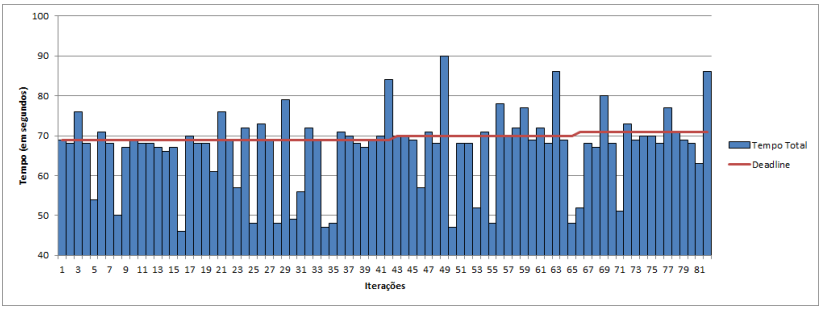
3.6. Resultados das Simulações

3.6.1. Simulação Mapa 1

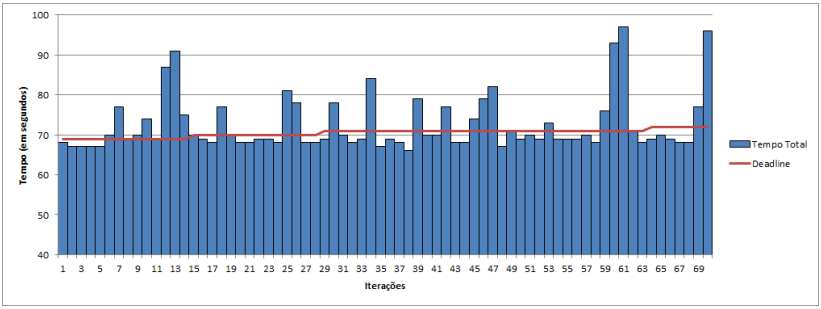
O *deadline* global definido para as simulações executadas neste mapa foi de 10 minutos. Nestas simulações, os robôs que atuam como obstáculos se movem aleatoriamente a cada intervalo de 45 segundos.

A quantidade de objetos carregados em cada uma das dez simulações em que o mecanismo é utilizado varia entre 28 e 30 objetos. Nas simulações em que as ações corretivas não são utilizadas, carregou-se 28 objetos. A Tabela 3.1 compara os resultados obtidos para a Tarefa 1 nas duas abordagens. Juntando as dez simulações, utilizando ações corretivas, a Tarefa 1 é executada 82 vezes, dentre elas, 58 (70,73%) são capazes de cumprir o *deadline* e 19 (32,75%), destas 58, são devido à utilização de ações corretivas. Sem estas ações, a Tarefa 1 é executada 70 vezes e seu *deadline* é cumprido 47 vezes (67,14%). A Figura 3.13 mostra uma comparação entre o tempo total que os robôs levam para completar as tarefas e o *deadline* de cada tarefa. A comparação da Figura 3.13 se refere aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial de histórico igual a 25 entradas.

Conforme os resultados apresentados na Tabela 3.2, nas simulações em que há a utilização de ações corretivas, na Tarefa 2, são executadas 85 iterações e 58 (68,23%) cumprem o *deadline*. Dentre estas 58 iterações, 22 (37,93%) ocorrem devido ao uso das ações corretivas. Sem estas ações, a Tarefa 2 é executada 71 vezes e cumpre o *deadline* em 49 (69,01%) dessas 71 iterações. Uma comparação entre o tempo total para completar uma tarefa e o *deadline* da tarefa é apresentada na Figura 3.14. Esta comparação se refere aos resultados obtidos para a



(a) Com ações corretivas



(b) Sem ações corretivas

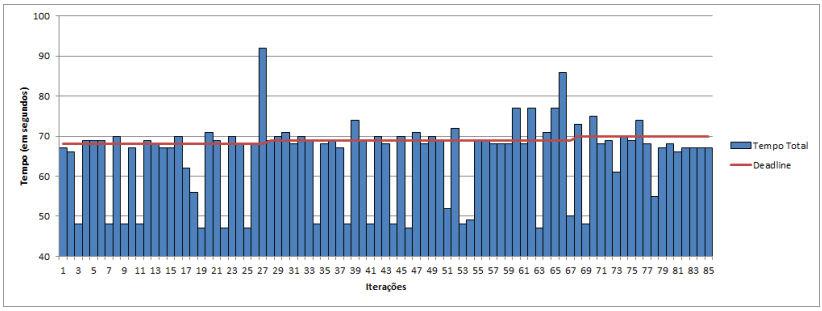
Figura 3.13: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 1

Tarefa 2 em ambas as abordagens e com o tamanho inicial de histórico igual a 25 entradas.

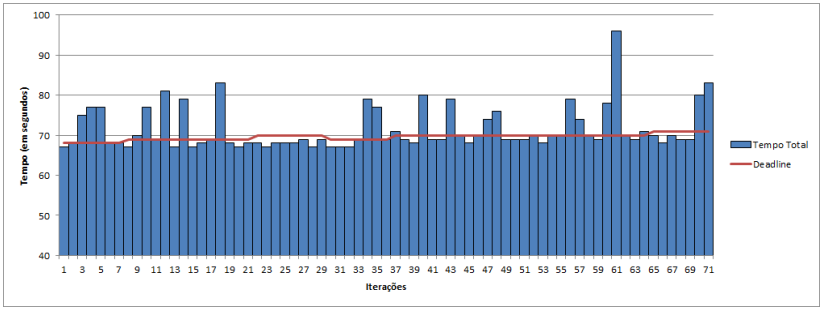
Dentre as simulações que utilizam ações corretivas, o valor do *deadline* varia entre 69 e 71 segundos na Tarefa 1 e entre 68 e 70 segundos na Tarefa 2. A média de objetos carregados por simulação é de 14,4 objetos para a Tarefa 1 e 14,7 objetos para a Tarefa 2. Nas simulações que não utilizam ações corretivas o *deadline* varia entre 69 e 72 segundos na Tarefa 1 e entre 68 e 71 segundos na Tarefa 2. A média de objetos carregados por simulação é de 14 objetos em ambas as tarefas. As Figuras 3.15a e 3.15b comparam a quantidade de objetos carregados para as tarefas 1 e 2, respectivamente, para o Mapa 1 e com tamanho inicial de histórico de 25 entradas.

Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	82	70
Cumpre *Deadline*	58 (70,73%)	47 (67,14%)
Não Cumpre *Deadline*	24 (29,27%)	23 (32,86%)
Cumpre *DL* c/ A. C.	19	-
Cumpre *DL* s/ A. C.	39	47

Tabela 3.1: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 1



(a) Com ações corretivas



(b) Sem ações corretivas

Figura 3.14: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 1

3.6.2. Simulação Mapa 2

O *deadline* global definido para as simulações executadas neste mapa é de 30 minutos. Nestas simulações, os robôs que atuam como obstáculos se movem aleatoriamente a cada intervalo de 60 segundos.

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	85	71
Cumpre *Deadline*	58 (68,23%)	49 (69,01%)
Não Cumpre *Deadline*	27 (31,77%)	22 (30,99%)
Cumpre *DL* c/ A. C.	22	-
Cumpre *DL* s/ A. C.	36	49

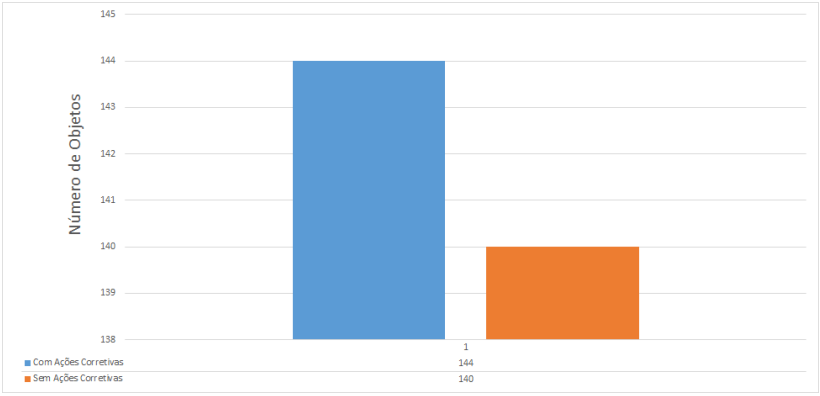
Tabela 3.2: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 1

Dentre as dez simulações realizadas em que ações corretivas são utilizadas, a quantidade de objetos carregados está no intervalo fechado entre 36 e 38 objetos, ao passo que são carregados de 32 a 38 objetos nas simulações que as ações corretivas não são utilizadas. Como apresentado na Tabela 3.3, que compara os valores obtidos para a Tarefa 1 nas duas abordagens, dentre todas as dez simulações nas quais as ações corretivas estão ativadas, a Tarefa 1 é executada 113 vezes e cumpre o *deadline* da tarefa em 69 (61,06%) iterações. Destas 69, 16 (23,18%) são devido à utilização de ações corretivas. Quando estas ações estão desabilitadas, a Tarefa 1 é executada 103 vezes e consegue respeitar as restrições temporais da tarefa 68 vezes (66,01%). Os gráficos da Figura 3.16 apresentam uma comparação entre o tempo total que os robôs levam para concluir as tarefas e o *deadline* de cada tarefa. Esta Figura compara os resultados obtidos na Tarefa 1 em ambas as abordagens e tendo um histórico com 25 entradas como histórico inicial.

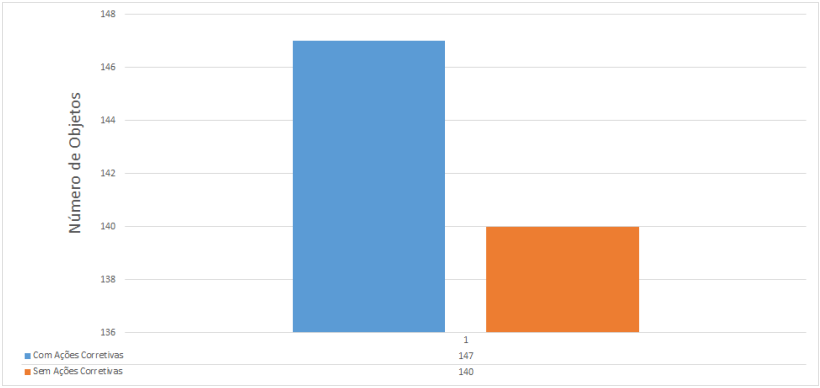
Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	113	103
Cumpre *Deadline*	69 (61,06%)	68 (66,01%)
Não Cumpre *Deadline*	44 (38,94%)	35 (33,99%)
Cumpre *DL* c/ A. C.	16	-
Cumpre *DL* s/ A. C.	53	68

Tabela 3.3: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 2

Quando as ações corretivas estão sendo utilizadas, a Tarefa 2 executa 109 iterações e o *deadline* é cumprido em 55 (50,45%) delas, como apresentado na Tabela 3.4, que compara os resultados da Tarefa 2 em ambas as abordagens. Destas 55 iterações, 50 (82,53%) conseguem cumprir o *deadline* devido ao uso das ações corretivas. Desabilitando



(a) Tarefa 1

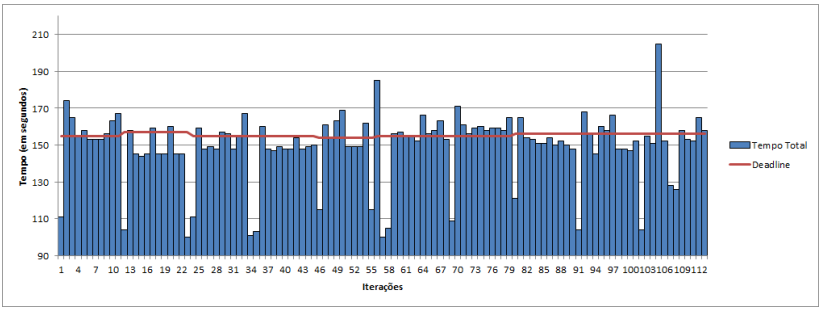


(b) Tarefa 2

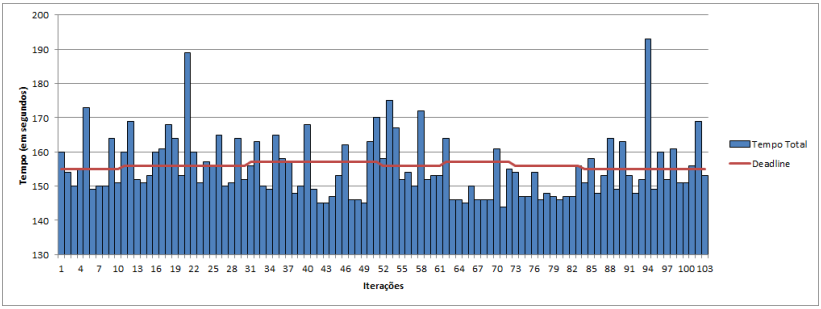
Figura 3.15: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 1

estas ações, a Tarefa 2 é executada 73 vezes e cumpre o *deadline* em 53 (72,6%) dessas 73 iterações. Na Figura 3.17, é apresentada, por meio de gráficos, uma comparação entre o valor do *deadline* de cada tarefa e o tempo que os robôs levam pra concluí-las em cada iteração. As comparações da Figura 3.17 se referem aos resultados obtidos para a Tarefa 2 em ambas as abordagens com o tamanho inicial de histórico igual a 25 entradas.

A variação no *deadline* das duas tarefas em simulações que utili-



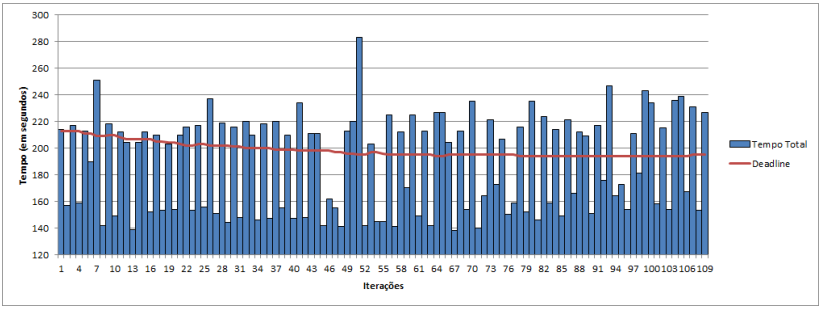
(a) Com ações corretivas



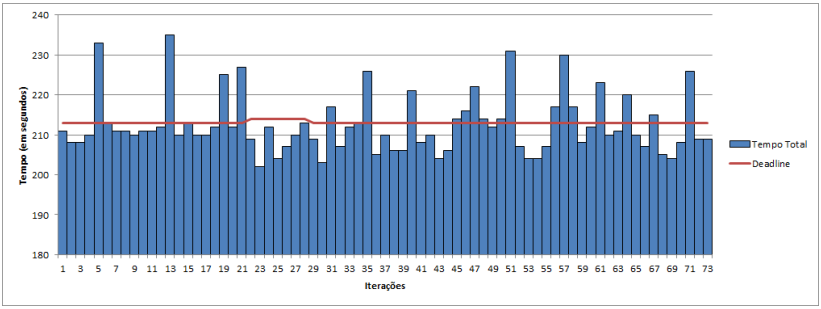
(b) Sem ações corretivas

Figura 3.16: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 2

zam ações corretivas assume valores entre 154 e 157 segundos na Tarefa 1 e entre 194 e 213 segundos na Tarefa 2. Quando as ações corretivas não estão sendo utilizadas, o valor do *deadline* varia entre 155 e 157 segundos na Tarefa 1 e entre 213 e 214 segundos na Tarefa 2. A média de objetos carregados por simulação apresenta resultados diferentes em cada abordagem, sendo que, para as simulações que utilizam ações corretivas, essa média é de 21 objetos para a Tarefa 1 e de 16,7 objetos para a Tarefa 2. Nas simulações que as ações corretivas estão desativadas, a média de objetos carregados é de 20,6 objetos para a Tarefa 1 e 14,6 objetos para a Tarefa 2. Uma comparação entre os objetos carregados no Mapa 2, com tamanho inicial de histórico de 25 entradas, é apresentada nas Figuras 3.18a e 3.18b para as tarefas 1 e 2 respectivamente.



(a) Com ações corretivas



(b) Sem ações corretivas

Figura 3.17: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 2

3.6.3. Simulação Mapa 3

Cada simulação executada neste cenário tem como *deadline* global o tempo de 30 minutos. Nestas simulações, os robôs que atuam como obstáculos se movem em momentos aleatórios a cada intervalo de 90 segundos.

Em cada uma das dez simulações realizadas, são carregados 14 objetos quando as ações corretivas são utilizadas e 12 objetos quando estas ações estão desativadas. Em todas as dez simulações que utilizam ações corretivas, conforme a Tabela 3.5, a Tarefa 1 executa 43 iterações e cumpre o *deadline* 27 vezes (62,79%), sendo que 16 (59,25%) ocorrem por consequência do uso de ações corretivas. As restrições temporais da Tarefa 1, nas simulações em que não há a utilização de ações corretivas, são respeitadas em 16 (61,9%) de suas 30 iterações. O tempo total que os robôs levam para completar as tarefas a cada iteração e o *deadline*

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	109	73
Cumpre *Deadline*	55 (50,45%)	53 (72,6%)
Não Cumpre *Deadline*	54 (49,55%)	20 (27,4%)
Cumpre *DL* c/ A. C.	50	-
Cumpre *DL* s/ A. C.	5	53

Tabela 3.4: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 2

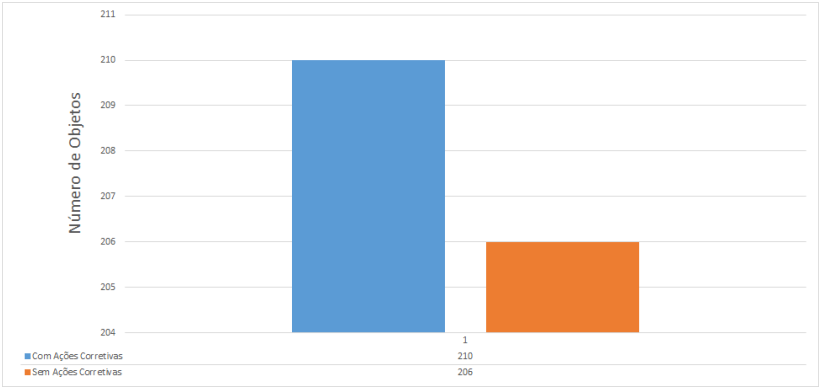
das tarefas são comparados na Figura 3.19. As comparações realizadas nesta figura apenas dizem respeito aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial do histórico de 25 entradas.

Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	43	30
Cumpre *Deadline*	27 (62,79%)	16 (53,34%)
Não Cumpre *Deadline*	16 (37,21%)	14 (46,66%)
Cumpre *DL* c/ A. C.	16	-
Cumpre *DL* s/ A. C.	11	16

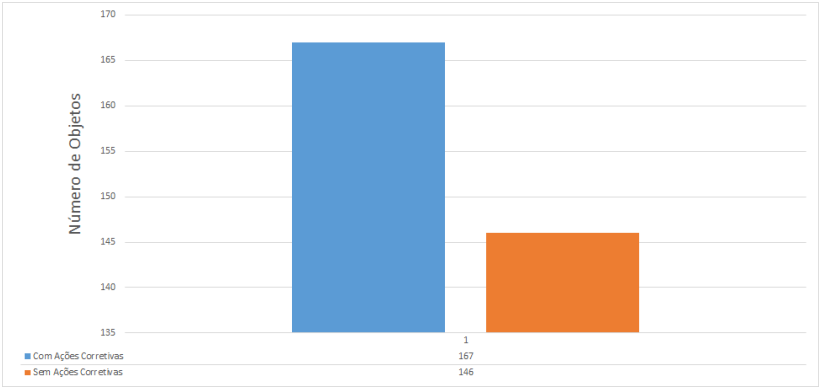
Tabela 3.5: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 3

A Tabela 3.6 também apresenta uma comparação entre os valores obtidos para a Tarefa 2 em ambas as abordagens. Esta tarefa é executada 41 vezes quando as ações corretivas estão habilitadas e 30 vezes quando elas não são utilizadas. Das 41 iterações da primeira abordagem, 30 (73,17%) conseguem cumprir o *deadline* e as ações corretivas são responsáveis por 12 (40%) dessas 30 iterações. Na segunda abordagem, 26 (86,67%) iterações são capazes de respeitar as restrições temporais da tarefa. Na Figura 3.20 são apresentados gráficos comparando o tempo que os robôs levam para cumprir as tarefas e o *deadline* cada iteração. Esta Figura compara os resultados obtidos para a Tarefa 2 nas duas abordagens e com um tamanho inicial de histórico igual a 25 entradas.

O valor do *deadline* da Tarefa 1 varia entre 491 e 497 segundos nas simulações em que as ações corretivas são utilizadas, enquanto que o da Tarefa 2 se mantém entre 498 e 501 segundos. Nestas simulações, a média de objetos carregados é de 7 em ambas as tarefas. Nas simulações



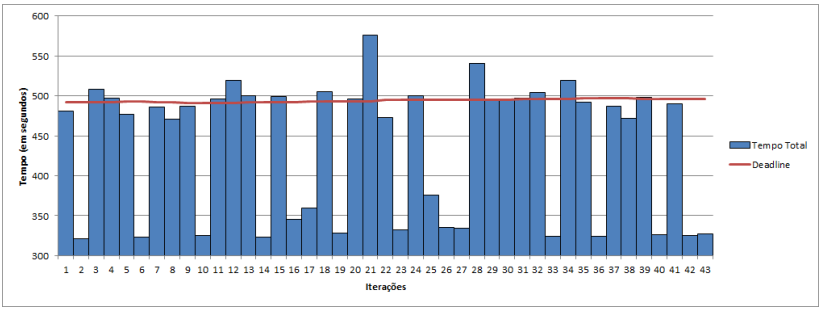
(a) Tarefa 1



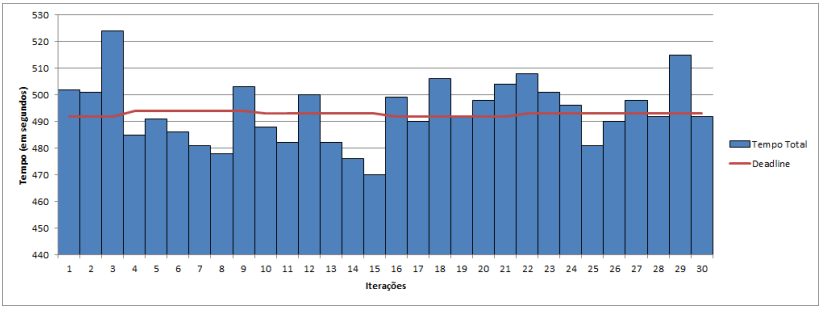
(b) Tarefa 2

Figura 3.18: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 2

realizadas sem as ações corretivas, o valor do *deadline* varia entre 492 e 494 segundos para a Tarefa 1 e situa-se no intervalo fechado entre 491 e 501 segundos para a Tarefa 2. A média de objetos carregados por simulação é de 6 objetos para ambas as tarefas. A quantidade de objetos carregados em ambas as abordagens, no Mapa 3, é comparada, para as tarefas 1 e 2, respectivamente, nos gráficos apresentados nas Figuras 3.21a e 3.21b. Estas figuras apresentam comparações para as simulações em que o tamanho inicial de histórico é igual a 25 entradas.



(a) Com ações corretivas



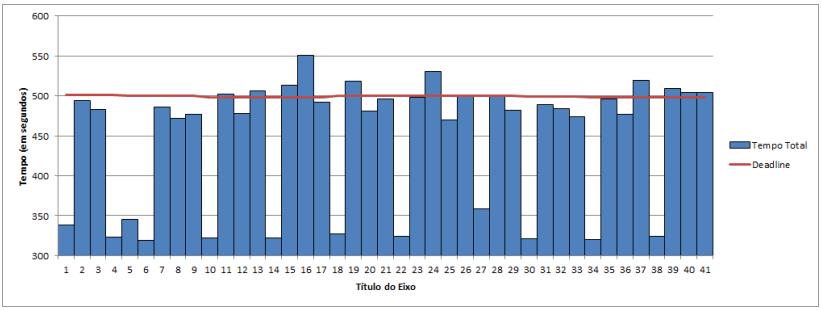
(b) Sem ações corretivas

Figura 3.19: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 3

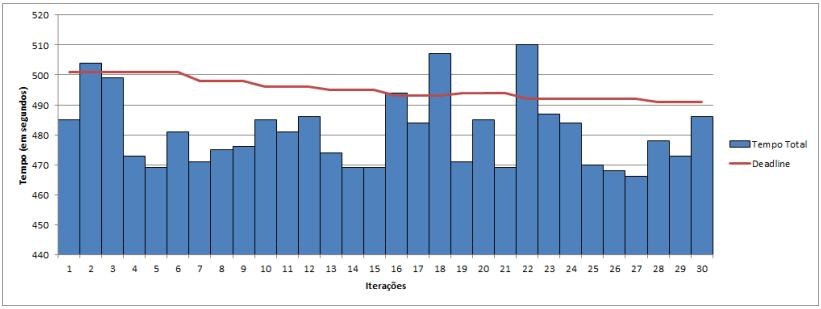
3.7. Conclusão

Este capítulo apresentou as ferramentas utilizadas na implementação do algoritmo, juntamente com a descrição do sistema e uma explicação sobre a execução das simulações. Os cenários nos quais as simulações foram executadas também são apresentados e os resultados obtidos em cada um deles foram discutidos.

Os resultados obtidos pelas simulações demonstram que a utilização das ações corretivas, decorrentes do resultado da previsão de perda de *deadline*, realizada pelo mecanismo, apresentam melhoras significativas quanto ao número de *deadlines* cumpridos, bem como quanto à quantidade de objetos carregados em cada simulação. Um exemplo de quão significativa pode ser essa melhora é apresentado no Mapa 3. Como nesse mapa cada tarefa leva cerca de oito minutos para ser completada, quando o robô não é obstruído por nenhum obstáculo, ser



(a) Com ações corretivas



(b) Sem ações corretivas

Figura 3.20: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 3

capaz de carregar dois objetos a mais dentro do intervalo de tempo definido pelo *deadline* global indica uma melhor eficiência na execução das tarefas, dado que foi possível carregar mais objetos do que sem a utilização das ações corretivas.

As ações corretivas propostas também comprovaram que são capazes de auxiliar o robô a cumprir o *deadline* neste modelo de tarefas. Devido ao uso destas ações, foi possível respeitar um número maior de restrições temporais das tarefas, bem como carregar um número maior de objetos comparado às tarefas que não utilizaram ações corretivas.

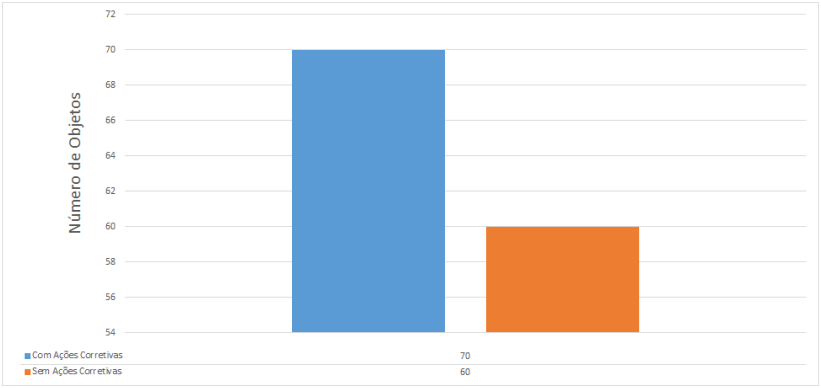
Nos três cenários, nas simulações que utilizaram ações corretivas, ambas as tarefas, na maioria dos casos, apresentaram uma taxa de não cumprimento de *deadline* superior a 20%. No terceiro mapa este valor é superior a 45%. Esta taxa de não cumprimento de *deadline* se deve à alta interferência dos obstáculos e, algumas vezes, do robô reserva, quando este sai para substituir algum dos outros robôs. Como

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	41	30
Cumpre *Deadline*	30 (73,17%)	26 (86,67%)
Não Cumpre *Deadline*	11 (26,83%)	4 (13,33%)
Cumpre *DL* c/ A. C.	12	-
Cumpre *DL* s/ A. C.	18	26

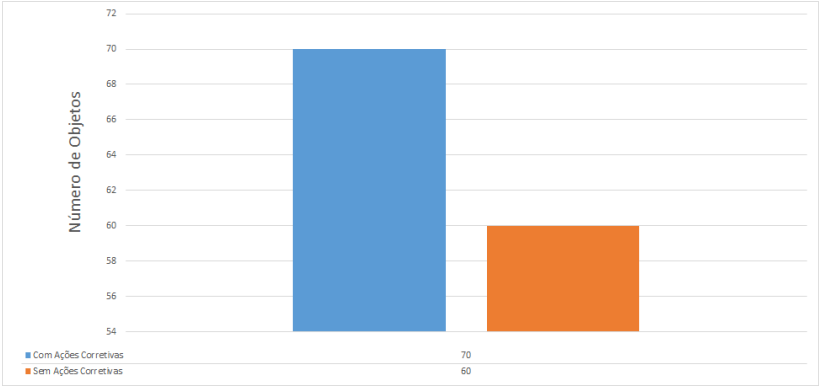
Tabela 3.6: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 3

o caminho realizado é quase o mesmo para que um robô substitua o outro, então há uma grande possibilidade que eles se encontrem no caminho e precisem desviar um do outro. É muito comum ambos os robôs desviarem para o mesmo lado, o que leva os dois a pararem completamente e, então, desviar apropriadamente. Como o *deadline* é calculado a partir dos dados presentes no histórico e estes dados são provenientes de execuções da tarefa na qual o robô não encontra nenhum obstáculo, apenas o fato de o robô ter que desviar de outro já pode ser o bastante para que ele perca o *deadline* da tarefa.

O próximo capítulo apresenta os trabalhos relacionados ao tema apresentado neste documento. Trabalhos referentes ao conceito de Árvores de Comportamento também são apresentados.



(a) Tarefa 1



(b) Tarefa 2

Figura 3.21: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 3

Capítulo 4

Trabalhos Relacionados

Este capítulo apresenta alguns trabalhos relacionados com o tema proposto neste documento. Os artigos citados são trabalhos na área da robótica que utilizam algum mecanismo de previsão para auxiliar no cumprimento de uma tarefa ou que precisam respeitar alguma restrição temporal durante a execução de uma atividade. Este capítulo possui duas seções. Na seção 4.1 é apresentada a forma através da qual os artigos foram buscados, quais as palavras chave utilizadas, as bases de dados onde a busca foi realizada e a quantidade total de trabalhos encontrados, em seguida, na seção 4.2 os artigos mais relevantes ao trabalho proposto são descritos. Uma conclusão para este capítulo é apresentada na seção 4.3.

4.1. Revisão Sistemática da Literatura

A busca por artigos que tratam da questão de pesquisa abordada neste trabalho foi realizada considerando as seguintes *strings*: ("*collaborative robots*" *OR* "*robot team*" *OR* "*task accomplishment prediction*" *OR* "*robot group*" *OR* "*robot collectives*" *OR* "*robot colony*" *OR* "*cooperative robots*") *AND* (("*deadline loss prediction*" *OR* "*time constraint*" *OR* "*time constraints*" *OR* "*time restriction*" *OR* "*time restrictions*") *OR* ("*prediction mechanism*" *OR* "*forecast mechanism*" *OR* "*prediction algorithm*" *OR* "*prediction technique*" *OR* "*forecast technique*")).

Foram considerados trabalhos publicados entre os anos de 2010 e 2015. As bases de dados utilizadas para realizar a busca dos artigos foram: *Science Direct*, *IEEE Xplore*, *Scopus* e *Web of Science*. A seleção dos artigos mais relevantes ocorreu em quatro etapas. Os resultados obtidos por cada etapa da seleção são apresentados na Figura 4.1. Ao fim do processo de seleção, quatro trabalhos foram escolhidos

e são apresentados a seguir.

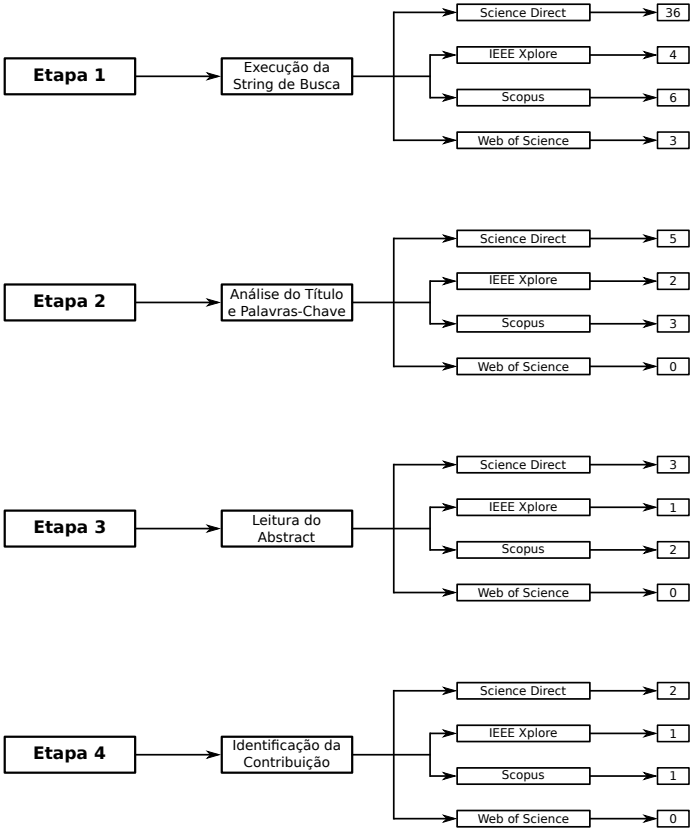


Figura 4.1: Etapas da Revisão de Literatura

4.2. Trabalhos Selecionados

O trabalho proposto por Carpin et al. [2014] aborda o envio de diversos robôs para objetivos definidos em um mapa conhecido. Os robôs devem chegar a todos os objetivos dentro de um determinado *deadline*. Neste artigo, os autores trabalham com um grande conjunto de pequenos robôs de baixo custo e com capacidades limitadas de sensoriamento e computação. Estes robôs não trocam nenhuma informação e a tarefa é considerada concluída quando todos os pontos objetivo tiverem sido alcançados por ao menos um robô do grupo. Processos de

Decisão de Markov Restritos (*Constrained Markov Decision Processes*) são usados para calcular estratégias que realizem um balanceamento entre segurança e velocidade para enviar os robôs aos pontos objetivo dentro da restrição temporal definida. Este trabalho pode ser utilizado em operações de busca e resgate em ambientes urbanos, vigilância e outras tarefas semelhantes.

Strom e Olson [2012] propõem um algoritmo para tratar do problema de predição de força de sinal quando todos os nós da rede estão explorando uma variedade de ambientes. O MATTE (***M**ulti-sensor **ATT**enuation **E**stimate*) realiza esta previsão se baseando na combinação de dados obtidos por outros sensores com as medidas de força de sinal para localizar obstáculos atenuantes no ambiente do robô. Portas, paredes e quaisquer outros obstáculos que possam estar entre os robôs podem atenuar a comunicação entre eles.

Guerrero e Oliver [2012] apresentaram um novo método de alocação de tarefas para grupos de robôs baseado em um algoritmo de leilão[1]. Este método é capaz de prever o tempo necessário para uma equipe de robôs completar uma tarefa, além de conseguir, também, determinar o tamanho do grupo para que as restrições temporais da tarefa sejam cumpridas. Neste artigo, também foi proposto o algoritmo de leilão no qual o método de alocação de tarefas se baseia, chamado de *Double Round auction*.

Em seu trabalho, Tardioli et al. [2015] propõem um protocolo para garantir comunicação em tempo real em uma rede "Ad hoc" de robôs. O RT-WMP (***R**eal-**T**ime **W**ireless **M**ulti-hop **P**rotocol*) é baseado em uma técnica de troca de *tokens*, possui suporte a prioridades e mobilidade e é capaz de garantir comunicação em tarefas com restrições firmes de tempo real.

A pesquisa realizada considerou grupos de robôs realizando tarefas com restrições temporais e mecanismos de previsão aplicados em grupos de robôs. O número de artigos encontrados nesta pesquisa foi relativamente baixo (quatro artigos) e, por isso, uma nova pesquisa foi realizada, considerando a aplicação de mecanismos de previsão em robôs móveis individuais. Os trabalhos encontrados nesta nova pesquisa mostraram estratégias desenvolvidas para perseguir e interceptar alvos móveis, planejamento de trajetória e para desviar de obstáculos. Em cada uma dessas aplicações, a predição apresentou bons resultados, como demonstrado nos trabalhos a seguir.

Zhu et al. [2013] tratam da dificuldade em perseguir e intercep-

[1] Abordagem na qual os robôs podem leiloar tarefas e executá-las [GERKEY; MATARI, 2002]

tar alvos móveis cuja trajetória e velocidade são desconhecidas e estão mudando dinamicamente, em ambientes com obstáculos. Para resolver esta questão, os autores propuseram um método que permite ao robô interceptar alvos móveis ao seguir diversas trajetórias curtas de linhas retas. O algoritmo procura calcular a velocidade, orientação e trajetória do alvo e prever um ponto para interceptá-lo. Caso o alvo altere sua trajetória, o ponto de interceptação será recalculado e a rota de navegação será replanejada. Este processo ocorre iteradamente até que o robô intercepte seu alvo. O algoritmo proposto por [ZHU et al., 2013] utiliza da previsão do ponto de interceptação para propor uma rota melhor que minimize o caminho percorrido pelo robô até a interceptação do alvo.

Miura et al. [1999] propõem um método de planejamento de movimento em situações em que haja obstáculos estáticos e móveis. Caso não exista comunicação entre o robô e os obstáculos, é necessário prever o movimento futuro destes para que seja possível planejar uma trajetória segura e eficiente. Este algoritmo calcula várias rotas possíveis e, para cada uma delas, estima o tempo que o robô levará para alcançar o destino. A rota que apresentar menor tempo será escolhida. Enquanto segue sua trajetória, o robô observa os obstáculos, estima sua posição e antecipa seu próximo movimento. Caso ele preveja que um obstáculo cruzará seu caminho, ele calcula se deve parar e esperar o obstáculo passar. Se isto for ocorrer, o robô verificará se esta rota ainda é mais rápida para ele alcançar o objetivo, levando em consideração o tempo em que ele ficará parado. Caso outra rota apresente um tempo menor, ele alterará sua trajetória. As simulações mostraram que o robô conseguiu atingir o destino em menos tempo ao escolher o caminho que minimizava o tempo de espera, mesmo se este fosse o caminho mais longo.

Shi et al. [2010] desenvolveram um método de desvio de obstáculos que utiliza predição para gerar sua trajetória. O algoritmo proposto neste artigo é baseado no LCM (*Lane Curvature Method*) e no BCM (*Beam Curvature Method*). O método proposto utiliza técnicas de previsão para auxiliar o robô a desviar de obstáculos móveis. Em testes realizados com obstáculos móveis, o BCM tradicional conseguiu chegar ao objetivo em apenas 11 das 30 tentativas, o LCM obteve sucesso em 16 tentativas e o algoritmo proposto (PBCM ou *Prediction based BCM*) conseguiu completar o percurso 28 vezes, o que comprova a sua eficiência quando comparado aos outros dois métodos citados.

Tsubouchi et al. [1993] também trabalharam em uma estratégia de desvio de obstáculos, em ambientes com uma grande concentração de

obstáculos móveis. Para desenvolver um método que apresentasse um bom desempenho neste tipo de ambiente, eles realizaram a união de um algoritmo iterativo de navegação para robôs móveis, que gera e utiliza uma previsão da movimentação dos obstáculos, com um algoritmo de aprendizado, que memoriza problemas no planejamento de trajetória junto com suas soluções para evitar recalculá-las. Neste algoritmo, o movimento de cada obstáculo é previsto assumindo-se que eles possuem velocidades constantes, este método é iterado frequentemente para se acomodar às variações de velocidade dos obstáculos.

Foi realizada uma busca por trabalhos que apresentassem a aplicação do conceito de Árvores de Comportamento em robôs móveis que utilizassem mecanismos de previsão. A *string* (*"Behavior Tree"*) *AND* (*"Mobile Robot"*) *AND* (*"Prediction Mechanism" OR "forecast mechanism" OR "prediction algorithm" OR "forecast algorithm" OR "prediction technique" OR "forecast technique"*) foi aplicada nas quatro bases de dados citadas anteriormente (*Science Direct*, *IEEE Xplore*, *Scopus* e *Web of Science*). A busca por artigos utilizando esta *string* não obteve resultados. Entretanto, alguns trabalhos que mostram a aplicação de Árvores de Comportamento na robótica móvel serão apresentados.

O trabalho de Bagnell et al. [2012] foi um dos primeiros a utilizar Árvore de Comportamento no processo de decisão de um robô, nomeada como **BART** (*Behavior Architecture for Robotic Tasks*). BART foi implementada como uma Árvore de Comportamento binária, na qual cada pai podia ter apenas dois filhos. Além dos nodos de sequência e seletor, BART possuía um nodo de controle paralelo, no qual todos os filhos eram executados simultaneamente.

Marzinotto et al. [2014] apresentam um *framework* de Árvore de Comportamento para controle robótico no qual a árvore é implementada como grafo acíclico direcionado, com nodos e folhas. Os tipos de nodo incluem os nodos raiz, seletor, sequência, paralelo e decorador como nodos de controle e os nodos de ação e condição como nodos de execução. Um nodo decorador é um nodo no qual o usuário pode mudar o retorno do filho baseado em uma condição interna do nodo decorador. Este nodo pode, também, ser utilizado como um mecanismo de sincronização quando é necessário controlar múltiplos agentes usando múltiplas árvores. Outra vantagem do *framework* proposto por Marzinotto é o *nodo**. Este nodo possui uma "memória" do estado anterior. Normalmente, uma Árvore de Comportamento é reiniciada cada vez que o nodo raiz reenvia o sinal. Se o intervalo entre os sinais é muito pequeno ou a árvore é muito grande, alguns nodos podem nunca ser executados, já que o sinal será enviado novamente antes que a árvore

possa executar todos os nodos. Para evitar este problema, o nodo* possui uma memória apontando para o último nodo executado pela BT, então, no próximo envio de sinal, se o nodo de controle é um nodo*, a árvore irá retomar a execução a partir do último nodo que recebeu o sinal. O nodo* é muito útil, especialmente para a robótica móvel, onde uma ação de movimentação pode durar mais tempo do que o intervalo entre os envios de sinais.

Colledanchise e Ogren [2014] usam a Árvore de Comportamento para modularizar robustez e segurança em sistemas híbridos, definindo um modelo funcional de diversos conceitos de uma Árvore de Comportamento tais como a forma de execução de uma BT, nodo de sequência, nodo seletor, dentre outros. Os autores definem segurança, eficiência e robustez de uma Árvore de Comportamento como modelos funcionais similares, fornecendo descrição teórica de como uma BT modulariza suas definições em sistemas robóticos.

Um outro trabalho relevante ao proposto neste documento é o de Monteiro [2014] que utiliza um mecanismo de previsão de perda de *deadline* para identificar se um robô móvel será capaz de completar uma tarefa respeitando as suas restrições temporais. Este trabalho foi apresentado na seção 2.3 deste documento.

Até onde foi buscado na literatura, o único trabalho que apresentou um mecanismo para prever se um robô será capaz ou não de cumprir uma tarefa dentro de seu *deadline* foi o de Monteiro [2014]. Guerrero e Oliver [2012] propõem um método para estimar o tempo que os robôs utilizarão para concluir a tarefa, mas este método não considera os obstáculos inesperados que podem aparecer em ambientes parcialmente conhecidos, ou seja, ambientes nos quais obstáculos desconhecidos podem ser adicionados.

A tabela apresentada na Figura 4.2 compara o trabalho apresentado neste documento com os quatro trabalhos relacionados mais relevantes, citados anteriormente. Os trabalhos foram comparados por meio da análise de quatro informações: tipo de ambiente; mecanismo de previsão utilizado; tipos de tarefas que podem ser realizadas e; se são consideradas restrições temporais.

Como apresentado na tabela da Figura 4.2, no trabalho de Carpin et al. [2014], o ambiente é conhecido, não é utilizado nenhum mecanismo de previsão, são consideradas as tarefas de busca e resgate e de vigilância e as restrições temporais são consideradas. Strom e Olson [2012] trabalharam com ambiente desconhecido, utilizaram mecanismo de previsão de perda de força de sinal, que é aplicado em tarefas de exploração de ambientes, entretanto este trabalho não considera res-

	Ambiente	Mecanismo de Previsão	Tarefa	Restrições temporais
Carpin et al. [2014]	Conhecido	–	Busca e resgate; Vigilância	X
Strom e Olson [2012]	Desconhecido	Perda de Força de Sinal	Exploração de Ambientes	–
Guerrero e Oliver [2012]	–	Tamanho do Grupo e Tempo de Execução	Deslocamento de Objetos	X
Tardioli et al. [2015]	–	–	Controle e coordenação de múltiplos robôs; Redes de sensores	X
Este Trabalho	**Parcialmente Conhecido**	**Perda de *Deadline***	**Deslocamento de Objetos**	**X**

Figura 4.2: Tabela comparando os trabalhos relacionados

trições temporais. Guerrero e Oliver [2012] não especificaram o tipo de ambiente no seu trabalho, o mecanismo de previsão proposto por eles é capaz de prever o tempo de execução da tarefa e tamanho do grupo de robôs, para que a tarefa possa ser cumprida dentro do prazo. Guerrero e Oliver [2012] trabalharam com a tarefa de deslocamento de objetos e consideraram as restrições temporais desta tarefa. O trabalho de Tardioli et al. [2015] também não deixa claro o tipo de ambiente e não utiliza mecanismos de previsão, o protocolo proposto pode ser utilizado para o controle e coordenação de múltiplos robôs ou em redes de sensores. Este protocolo considera as restrições temporais.

A proposta deste trabalho pode ser executada em ambientes parcialmente conhecidos, ou seja, ambientes nos quais obstáculos desconhecidos podem ser encontrados, o mecanismo de previsão adotado foi o mecanismo de previsão de perda de *deadline*, a tarefa realizada pelos robôs é o deslocamento de objetos e as restrições temporais da tarefa são consideradas.

4.3. Conclusão

Neste capítulo foram apresentados alguns trabalhos relacionados ao tema proposto neste documento. O próximo capítulo traz as considerações finais deste trabalho, bem como propostas de trabalhos futuros que podem ser realizados a partir do proposto neste documento.

Capítulo 5

Considerações Finais

Este trabalho abordou a utilização de ações corretivas por robôs móveis executando tarefas que apresentassem restrições temporais. Estas ações são executadas quando o mecanismo de previsão de perda de *deadline* prevê que o robô não será capaz de completar a tarefa dentro do prazo. Também foi abordado, neste trabalho, a utilização da Árvore de Comportamento em sistemas robóticos, um assunto ainda pouco abordado na literatura de robótica móvel.

Para o desenvolvimento deste trabalho, foi realizada uma revisão bibliográfica com o intuito de auxiliar na identificação de trabalhos cujo escopo fosse semelhante ao desenvolvido neste documento. Apesar de poucos trabalhos relacionados ao tema proposto terem sido encontrados, eles foram de grande uso para aprender sobre algumas estratégias utilizadas para a resolução de problemas que apresentassem restrições temporais, bem como mecanismos de previsão utilizados em grupos de robôs.

O objetivo geral definido para este trabalho foi alcançado e os resultados obtidos estão de acordo com o previsto. Os objetivos específicos, definidos com o intuito de alcançar os objetivos gerais, também foram concluídos com êxito. O primeiro objetivo específico tratou da adaptação do mecanismo de previsão de perda de *deadline* proposto por Monteiro [2014] para o *Robot Operating System* (*ROS*). Este mesmo mecanismo teve suas funções divididas em nodos para que, dessa forma, pudesse ser implementado utilizando o conceito de Árvore de Comportamento, concluindo o segundo objetivo específico.

O terceiro objetivo consistia na realização de testes, nos quais as ações corretivas desenvolvidas foram aplicadas em um grupo de robôs móveis, enquanto realizavam tarefas de deslocamento de objetos. Estes testes foram realizados com o propósito de validar e analisar o con-

junto de ações corretivas proposto. Para a conclusão deste objetivo, foi utilizado o simulador *Stage*, definiu-se um grupo de robôs móveis e cada um deles utilizava o mecanismo de previsão enquanto executava tarefas. Desenvolveu-se, também, um nodo responsável pela troca de mensagens entre os robôs do grupo, o nodo *Linker*. Diversas simulações foram realizadas e apresentaram resultados positivos, comprovando que a utilização das ações corretivas permitiu que os robôs respeitassem as restrições temporais das tarefas em mais execuções, quando comparado às simulações que não utilizavam as ações corretivas. A utilização destas ações de recuperação também permitiram que os robôs carregassem uma maior quantidade de objetos por simulação, o que reforça a vantagem na utilização das ações corretivas.

Resultados preliminares obtidos neste trabalho permitiram a publicação do artigo *Using a Deadline Missing Prediction Mechanism and Recovery Actions to Avoid Deadline Losses* [RAIA et al., 2016] no evento *International Conference on Computers and Their Applications* (*CATA*), que ocorreu de quatro a seis de abril de 2016. Um artigo que apresenta os resultados finais, descritos neste trabalho, está sendo escrito para, então, ser publicado em um periódico qualificado na área de Ciência da Computação.

Este trabalho mostrou que a utilização das ações corretivas é essencial em tarefas que têm o tempo como restrição. Utilizando estas ações, os robôs conseguiram se recuperar de prováveis perdas de *deadline* e também foram capazes de obter melhor desempenho na realização das tarefas, ao carregarem mais objetos nas simulações em que as ações corretivas estavam habilitadas.

5.1. Trabalhos Futuros

A seguir são citados alguns exemplos de trabalhos futuros que podem ser desenvolvidos:

- Ampliar o mecanismo proposto para que as previsões sejam realizadas periodicamente durante a navegação, analisando as alterações que podem ocorrer no ambiente;
- Analisar o consumo energético dos robôs, para verificar se a utilização das ações corretivas apresenta um consumo que inviabilizaria seu uso;
- Desenvolver um conjunto mais abrangente de ações corretivas, ou seja, que é adequado para diferentes tipos de tarefa;
- Realizar testes em robôs reais, para verificar se os resultados obtidos nas simulações se repetem em ambientes reais.

Referências

BAGNELL, J. A.; CAVALCANTI, F.; CUI, L.; GALLUZZO, T.; HEBERT, M.; KAZEMI, M.; KLINGENSMITH, M.; LIBBY, J.; LIU, T. Y.; POLLARD, N. et al. An integrated system for autonomous robotics manipulation. In: IEEE. **Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on**. [S.l.], 2012. p. 2955–2962.

CAI, B.; HUANG, S.; LIU, D.; DISSANAYAKE, G. Rescheduling policies for large-scale task allocation of autonomous straddle carriers under uncertainty at automated container terminals. **Robotics and Autonomous Systems**, Elsevier B.V., v. 62, n. 4, p. 506–514, 2014. ISSN 09218890. Disponível em: <http://dx.doi.org/10.1016/j.robot.2013.12.007>.

CARPIN, S.; PAVONE, M.; SADLER, B. M. Rapid Multirobot Deployment with Time Constraints. In: **IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS**. Chicago, IL, USA: [s.n.], 2014. p. 1147–1154. ISBN 9781479969340.

COLLEDANCHISE, M.; OGREN, P. How behavior trees modularize robustness and safety in hybrid systems. In: **Intelligent Robots and Systems (IROS)**. [S.l.: s.n.], 2014.

FARINES, J.-M.; FRAGA, J. d. S.; OLIVEIRA, R. d. Sistemas de tempo real. **Escola de Computação**, v. 2000, p. 201, 2000.

GERKEY, B. P.; MATARI, M. J. Sold!: Auction methods for multirobot coordination. **Robotics and Automation, IEEE Transactions on**, IEEE, v. 18, n. 5, p. 758–768, 2002.

GUERRERO, J.; OLIVER, G. Multi-robot coalition formation in real-time scenarios. **Robotics and Autonomous Systems**, Elsevier B.V., v. 60, n. 10, p. 1295–1307, 2012. ISSN 09218890. Disponível em: <http://dx.doi.org/10.1016/j.robot.2012.06.004>.

ISLA, D. Handling complexity in the halo 2 ai. In: **Game Developers Conference**. [S.l.: s.n.], 2005. v. 12.

JIA, M. J. M.; ZHOU, G. Z. G.; CHEN, Z. C. Z. An efficient strategy integrating grid and topological information for robot exploration. **IEEE Conference on Robotics, Automation and Mechatronics, 2004.**, v. 2, p. 1–3, 2004.

MARINO, A.; PARKER, L.; ANTONELLI, G.; CACCAVALE, F. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In: IEEE. **Robotics and Automation, 2009. ICRA'09. IEEE International Conference on**. [S.l.], 2009. p. 831–836.

MARZINOTTO, A.; COLLEDANCHISE, M.; SMITH, C.; OGREN, P. Towards a unified behavior trees framework for robot control. In: **Robotics and Automation (ICRA), 2014 IEEE International Conference**. [S.l.: s.n.], 2014.

MEI, Y.; LU, Y. H.; LEE, C. S. G.; HU, Y. C. Energy-efficient mobile robot exploration. **Proceedings - IEEE International Conference on Robotics and Automation**, v. 2006, n. May, p. 505–511, 2006. ISSN 10504729.

MIURA, J.; UOZUMI, H.; SHIRAI, Y. Mobile robot motion planning considering the motion uncertainty of moving obstacles. **IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics**, v. 4, p. 692–697, 1999. ISSN 1062-922X.

MONTEIRO, E. R. d. B. **Mecanismo de Previsão de Perda de Deadline para a Navegação de Robôs Móveis Autônomos**. 109 p. Dissertação (Mestrado) — Programa de Pós-Graduação em Ciência da Computação (PPGCC), Universidade Federal de Santa Catarina, Florianópolis, 2014.

O'KANE, J. M. **A Gentle Introduction to ROS**. [s.n.], 2013. ISBN

978-1492143239. Disponível em: <http://www.cse.sc.edu/~jokane/agitr/>.

RAIA, H. F.; SIQUEIRA, F. d. L.; PLENTZ, P. D. M. Using a deadline missing prediction mechanism and recovery actions to avoid deadline losses. In: **Proceedings of the 31th International Conference on Computers and their Applications, Las Vegas, Nevada**. [S.l.: s.n.], 2016.

SHI, C.; WANG, Y.; YANG, J. A local obstacle avoidance method for mobile robots in partially known environment. **Robotics and Autonomous Systems**, Elsevier B.V., v. 58, n. 5, p. 425–434, 2010. ISSN 09218890. Disponível em: <http://dx.doi.org/10.1016/j.robot.2010.02.005>.

SIQUEIRA, F. d. L.; PIERI, E. R. D. A context-aware approach to the navigation of mobile robots. In: **Simposio Brasileiro de Automacao Inteligente SBAI 2015**. [S.l.: s.n.], 2015.

STROM, J.; OLSON, E. Multi-sensor ATTenuation Estimation (MATTE): Signal-strength prediction for teams of robots. In: **IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2012. p. 4730–4736. ISBN 9781467317375. ISSN 21530858.

TARDIOLI, D.; SICIGNANO, D.; VILLARROEL, J. L. A wireless multi-hop protocol for real-time applications. **Computer Communications**, Elsevier B.V., v. 55, p. 4–21, 2015. ISSN 01403664. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0140366414003028>.

TSUBOUCHI, T.; HIROSE, a.; ARIMOTO, S. A navigation scheme with learning for a mobile robot among multiple moving obstacles. **Proceedings of 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '93)**, v. 3, n. C, p. 2234–2240, 1993.

ZHU, Q.; HU, J.; HENSCHEN, L. A new moving target interception algorithm for mobile robots based on sub-goal forecasting and an improved scout ant algorithm. **Applied Soft Computing Journal**, Elsevier B.V., v. 13, n. 1, p. 539–549, 2013. ISSN 15684946. Disponível em: <http://dx.doi.org/10.1016/j.asoc.2012.08.013>.

Apêndice A

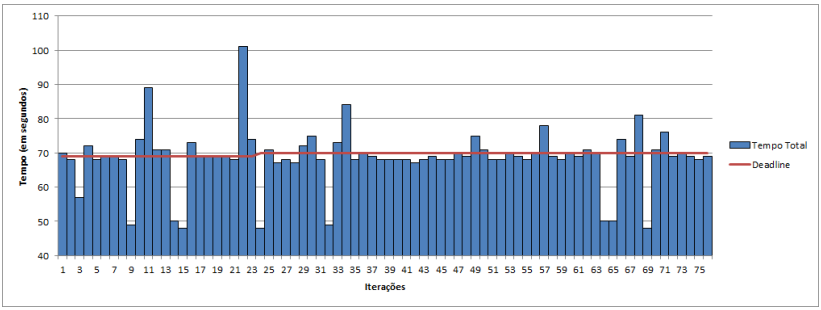
Resultado das Simulações com Histórico Inicial com 50 Entradas

A.1. Mapa 1

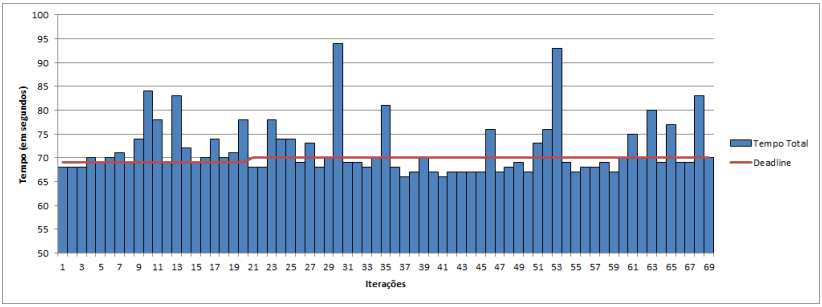
Dentre as dez simulações realizadas em que ações corretivas são utilizadas, a quantidade de objetos carregados está no intervalo fechado entre 28 e 31 objetos, ao passo que são carregados de 26 a 28 objetos nas simulações que as ações corretivas não são utilizadas. Como apresentado na Tabela A.1, que compara os valores obtidos para a Tarefa 1 nas duas abordagens, dentre todas as dez simulações nas quais as ações corretivas estão ativadas, a Tarefa 1 é executada 76 vezes e cumpre o *deadline* da tarefa em 54 (71,05%) iterações. Destas 54, nove (16,67%) são devido à utilização de ações corretivas. Quando estas ações estão desabilitadas, a Tarefa 1 é executada 69 vezes e consegue respeitar as restrições temporais da tarefa 42 vezes (60,86%). Os gráficos da Figura A.1 apresentam uma comparação entre o tempo total que os robôs levam para concluir as tarefas e o *deadline* de cada tarefa. Esta Figura compara os resultados obtidos na Tarefa 1 em ambas as abordagens e tendo um histórico com 50 entradas como histórico inicial.

Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	76	69
Cumpre *Deadline*	54 (71,05%)	42 (60,86%)
Não Cumpre *Deadline*	22 (28,95%)	27 (39,14%)
Cumpre *DL* c/ A. C.	9	-
Cumpre *DL* s/ A. C.	45	42

Tabela A.1: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 1



(a) Com ações corretivas

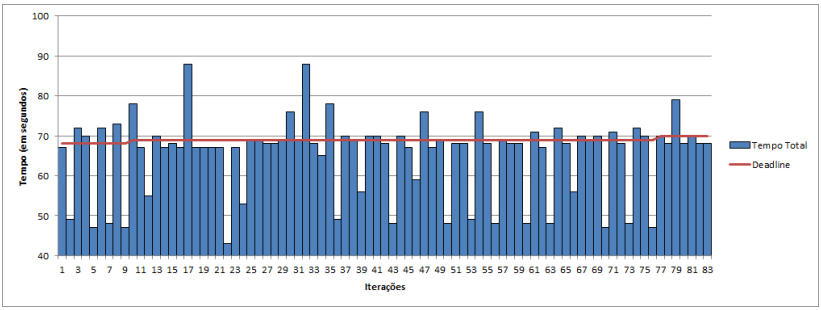


(b) Sem ações corretivas

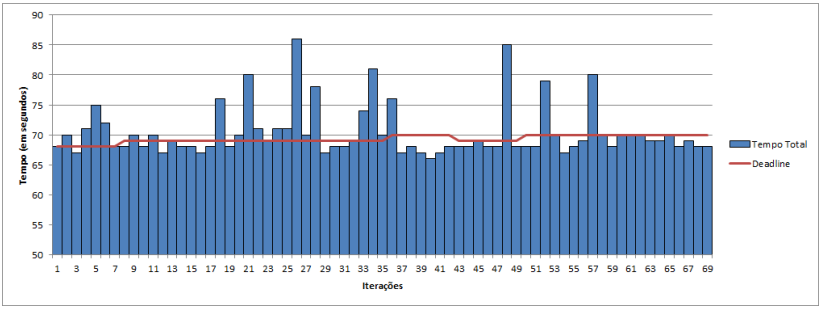
Figura A.1: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 1

Quando as ações corretivas estavam sendo utilizadas, a Tarefa 2 executa 83 iterações e o *deadline* é cumprido em 59 (71,08%) delas, como apresentado na Tabela A.2, que compara os resultados da Tarefa 2 em ambas as abordagens. Destas 59 iterações, 19 (32,2%) conseguem cumprir o *deadline* devido ao uso das ações corretivas. Desabilitando estas ações, a Tarefa 2 é executada 69 vezes e cumpre o *deadline* em 47 (68,11%) dessas 69 iterações. Na Figura A.2, é apresentada, por meio de gráficos, uma comparação entre o valor do *deadline* de cada tarefa e o tempo que os robôs levaram pra concluí-las em cada iteração. As comparações da Figura A.2 se referem aos resultados obtidos para a Tarefa 2 em ambas as abordagens com o tamanho inicial de histórico igual a 50 entradas.

A variação no *deadline* das duas tarefas em simulações que não utilizam ações corretivas assumiu os mesmos valores de quando as ações corretivas são utilizadas, ou seja, variou entre 69 e 70 segundos na Ta-



(a) Com ações corretivas



(b) Sem ações corretivas

Figura A.2: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 1

refa 1 e entre 68 e 70 segundos na Tarefa 2. Entretanto a média de objetos carregados por simulação apresenta resultados diferentes em cada abordagem, sendo que, para as simulações que utilizam ações corretivas, essa média é de 14,1 objetos para a Tarefa 1 e de 13,4 objetos para a Tarefa 2. Nas simulações que as ações corretivas estavam desativadas, a média de objetos carregados é de 13,8 objetos para ambas as tarefas. Uma comparação entre os objetos carregados no Mapa 1, com tamanho inicial de histórico de 50 entradas, é apresentada nas Figuras A.3a e A.3b para as tarefas 1 e 2 respectivamente.

A.2. Mapa 2

Em cada uma das dez simulações realizadas, são carregados entre 36 e 38 objetos quando as ações corretivas são utilizadas e entre 34 e 36 objetos quando estas ações estão desativadas. Em todas as dez simu-

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	83	69
Cumpre *Deadline*	59 (71,08%)	47 (68,11%)
Não Cumpre *Deadline*	24 (28,92%)	22 (31,89%)
Cumpre *DL* c/ A. C.	19	-
Cumpre *DL* s/ A. C.	40	47

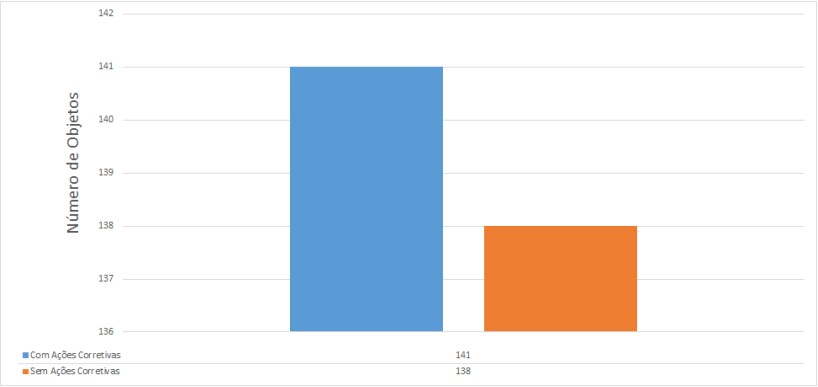
Tabela A.2: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 1

lações que utilizam ações corretivas, conforme a Tabela A.3, a Tarefa 1 executa 114 iterações e cumpre o *deadline* 78 vezes (68,42%), sendo que 14 (17,94%) ocorrem por consequência do uso de ações corretivas. As restrições temporais da Tarefa 1, nas simulações em que não há a utilização de ações corretivas, são respeitadas em 74 (73,26%) de suas 101 iterações. O tempo total que os robôs levam para completar as tarefas a cada iteração e o *deadline* das tarefas são comparados na Figura A.4. As comparações realizadas nesta figura apenas dizem respeito aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial do histórico de 50 entradas.

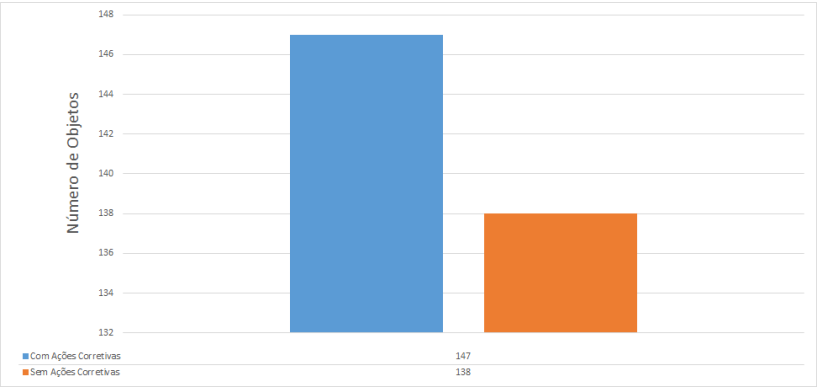
Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	114	101
Cumpre *Deadline*	78 (68,42%)	74 (73,26%)
Não Cumpre *Deadline*	36 (31,58%)	27 (26,74%)
Cumpre *DL* c/ A. C.	14	-
Cumpre *DL* s/ A. C.	64	74

Tabela A.3: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 2

A Tabela A.4 também apresenta uma comparação entre os valores obtidos para a Tarefa 2 em ambas as abordagens. Esta tarefa é executada 105 vezes quando as ações corretivas estão habilitadas e 76 vezes quando elas não são utilizadas. Das 105 iterações da primeira abordagem, 63 (60%) conseguem cumprir o *deadline* e as ações corretivas são responsáveis por 52 (82,53%) dessas 63 iterações. Na segunda abordagem, 55 (72,36%) iterações são capazes de respeitar as restrições temporais da tarefa. Na Figura A.5 são apresentados gráficos comparando o tempo que os robôs levam para cumprir as tarefas e o *deadline* cada iteração. Esta Figura compara os resultados obtidos para a Tarefa



(a) Tarefa 1

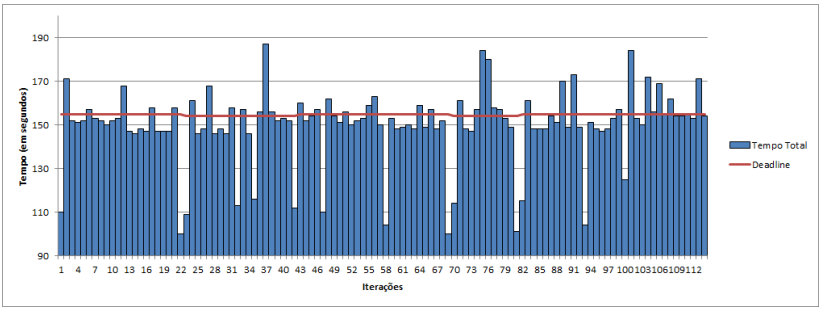


(b) Tarefa 2

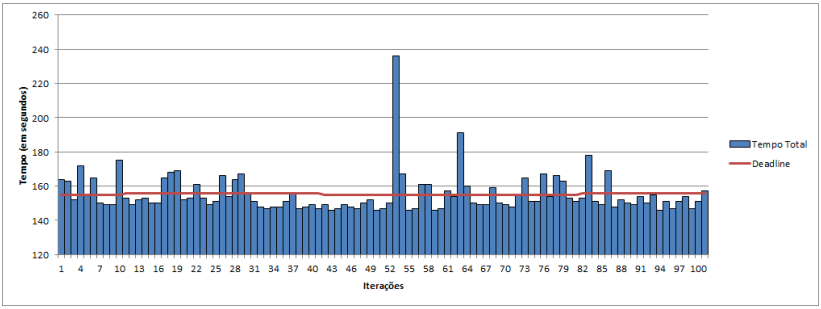
Figura A.3: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 1

2 nas duas abordagens e com um tamanho inicial de histórico igual a 50 entradas.

O valor do *deadline* da Tarefa 1 varia entre 154 e 155 segundos nas simulações em que as ações corretivas são utilizadas, enquanto que o da Tarefa 2 se mantém entre 195 e 214 segundos. Nestas simulações, a média de objetos carregados é de 21,2 objetos para a Tarefa 1 e de 15,8 objetos para a Tarefa 2. Nas simulações realizadas sem as ações corretivas, o valor do *deadline* varia entre 155 e 156 segundos para a Tarefa 1 e situa-se no intervalo fechado entre 213 e 214 segundos para a



(a) Com ações corretivas



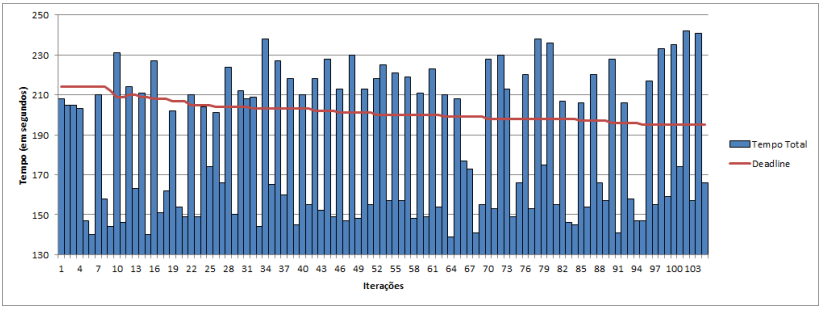
(b) Sem ações corretivas

Figura A.4: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 2

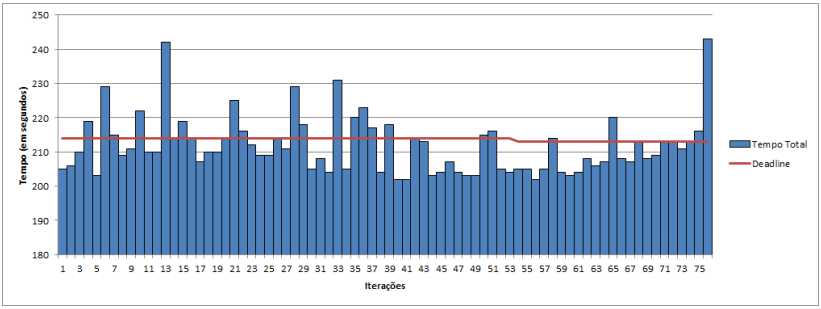
Tarefa 2. A média de objetos carregados por simulação é de 20,2 objetos para a Tarefa 1 e de 15,2 objetos para a Tarefa 2. A quantidade de objetos carregados em ambas as abordagens, no Mapa 2, é comparada, para as tarefas 1 e 2, respectivamente, nos gráficos apresentados nas Figuras A.6a e A.6b. Estas figuras apresentam comparações para as simulações em que o tamanho inicial de histórico é igual a 50 entradas.

A.3. Mapa 3

A quantidade de objetos carregados em cada uma das dez simulações em que o mecanismo é utilizado é de 14 objetos. Nas simulações em que as ações corretivas não são utilizadas, carregou-se 12 objetos. A Tabela A.5 compara os resultados obtidos para a Tarefa 1 nas duas abordagens. Juntando as dez simulações, utilizando ações corretivas, a Tarefa 1 é executada 43 vezes, dentre elas, 30 (69,76%) são capazes



(a) Com ações corretivas



(b) Sem ações corretivas

Figura A.5: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 2

de cumprir o *deadline* e 16 (53,34%), destas 30, são devido à utilização de ações corretivas. Sem estas ações, a Tarefa 1 é executada 30 vezes e seu *deadline* é cumprido 19 vezes (63,34%). A Figura A.7 mostra uma comparação entre o tempo total que os robôs levaram para completar as tarefas e o *deadline* de cada tarefa. A comparação da Figura A.7 se refere aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial de histórico igual a 50 entradas.

Conforme os resultados apresentados na Tabela A.6, nas simulações em que há a utilização de ações corretivas, na Tarefa 2, são executadas 42 iterações e 31 (73,8%) cumprem o *deadline*. Dentre estas 31 iterações, 14 (45,16%) ocorrem devido ao uso das ações corretivas. Sem estas ações, a Tarefa 2 é executada 30 vezes e cumpre o *deadline* em 24 (80%) dessas 30 iterações. Uma comparação entre o tempo total para completar uma tarefa e o *deadline* da tarefa é apresentada na Figura A.8. Esta comparação se refere aos resultados obtidos para a Tarefa 2

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	105	76
Cumpre *Deadline*	63 (60%)	55 (72,36%)
Não Cumpre *Deadline*	42 (40%)	21 (27,64%)
Cumpre *DL* c/ A. C.	52	-
Cumpre *DL* s/ A. C.	11	55

Tabela A.4: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 2

Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	43	30
Cumpre *Deadline*	30 (69,76%)	19 (63,34%)
Não Cumpre *Deadline*	13 (30,24%)	11 (36,66%)
Cumpre *DL* c/ A. C.	16	-
Cumpre *DL* s/ A. C.	14	19

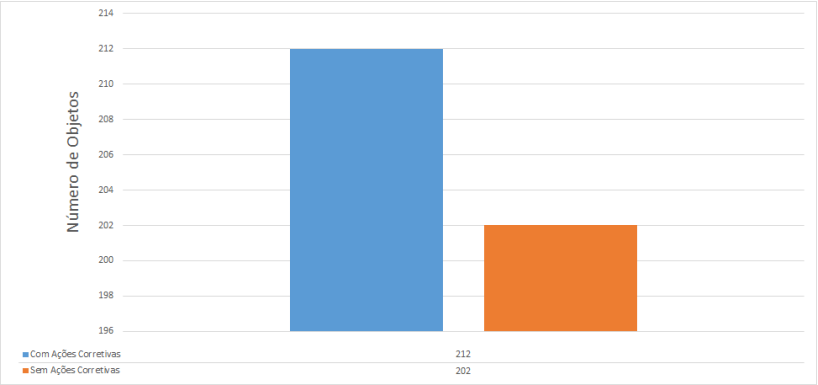
Tabela A.5: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 3

em ambas as abordagens e com o tamanho inicial de histórico igual a 50 entradas.

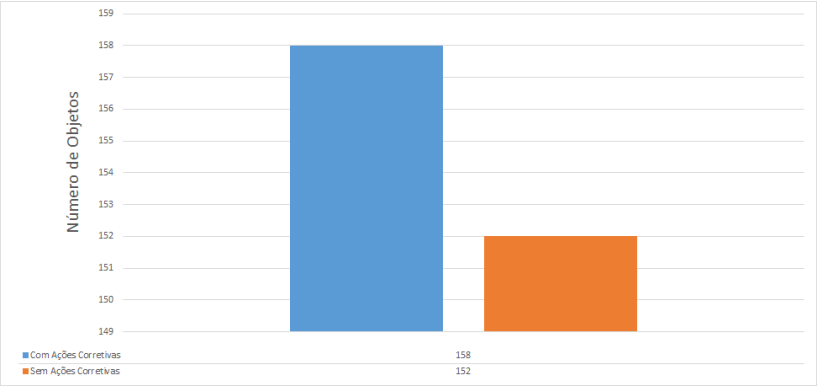
Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	42	30
Cumpre *Deadline*	31 (73,8%)	24 (80%)
Não Cumpre *Deadline*	11 (26,2%)	6 (20%)
Cumpre *DL* c/ A. C.	14	-
Cumpre *DL* s/ A. C.	17	24

Tabela A.6: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 3

Dentre as simulações que utilizam ações corretivas, o valor do *deadline* varia entre 492 e 494 segundos na Tarefa 1 e entre 495 e 497 segundos na Tarefa 2. A média de objetos carregados por simulação é de 7 objetos para as duas tarefas. Nas simulações que não utilizam ações corretivas o *deadline* varia entre 492 e 495 segundos na Tarefa 1 e entre 493 e 496 segundos na Tarefa 2. A média de objetos carregados por simulação é de 6 objetos em ambas as tarefas. As Figuras A.9a e A.9b comparam a quantidade de objetos carregados para as tarefas 1 e



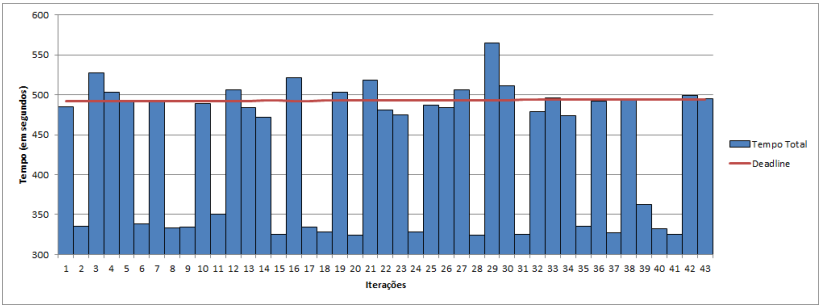
(a) Tarefa 1



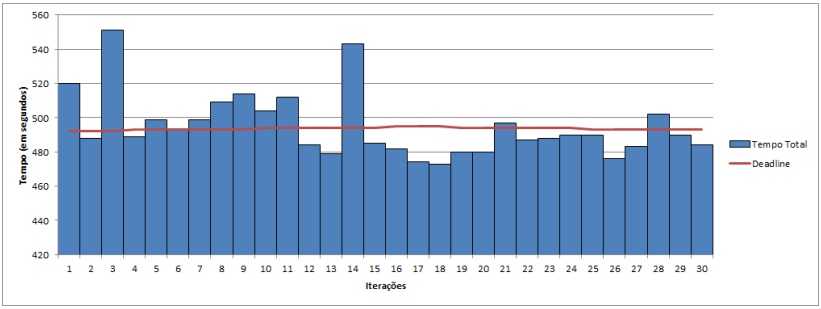
(b) Tarefa 2

Figura A.6: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 2

2, respectivamente, para o Mapa 3 e com tamanho inicial de histórico de 50 entradas.

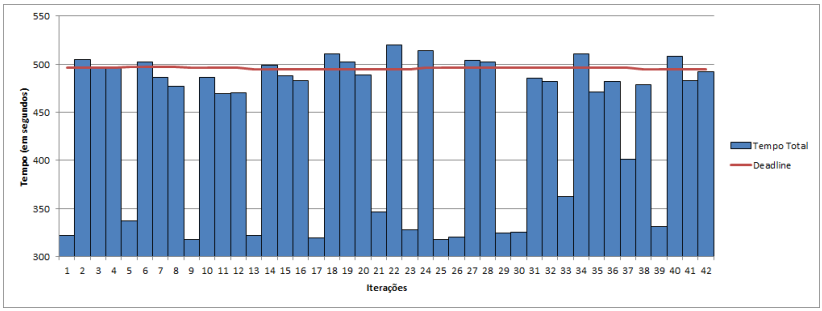


(a) Com ações corretivas

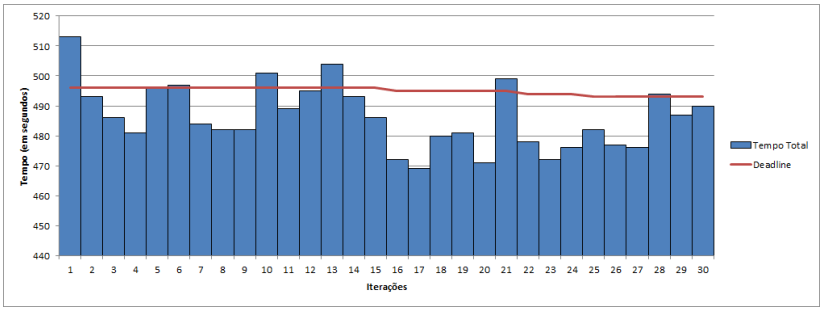


(b) Sem ações corretivas

Figura A.7: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 3

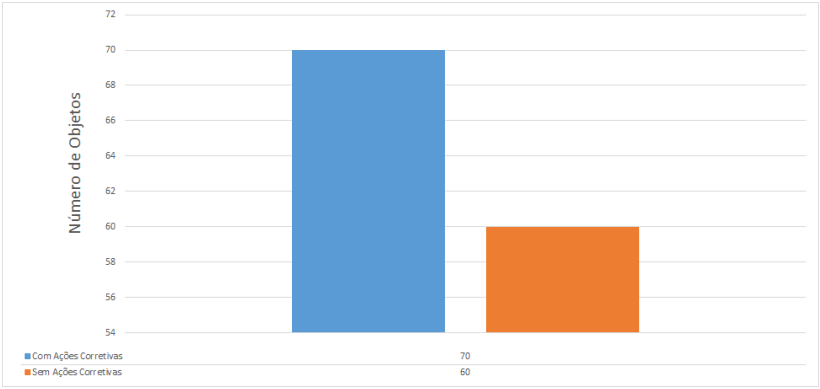


(a) Com ações corretivas

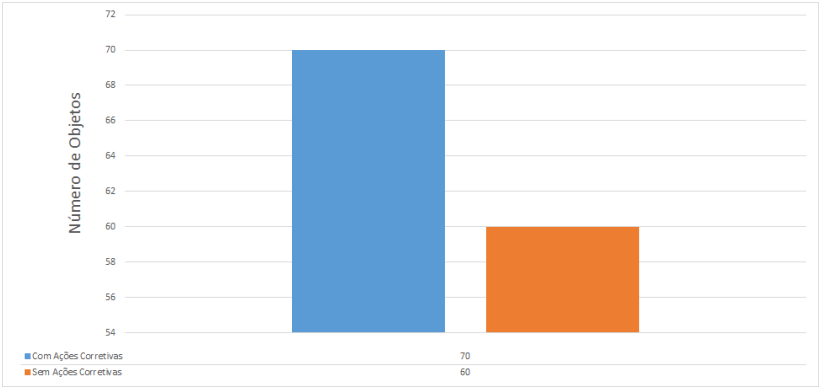


(b) Sem ações corretivas

Figura A.8: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 3



(a) Tarefa 1



(b) Tarefa 2

Figura A.9: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 3

Apêndice B

Resultado das Simulações com Histórico Inicial com 100 Entradas

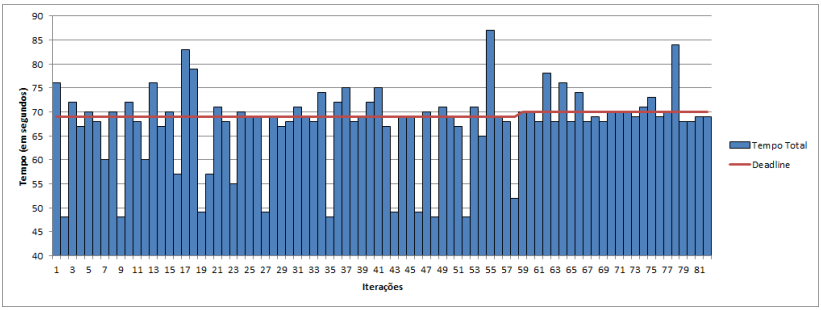
B.1. Mapa 1

Em cada uma das dez simulações realizadas, são carregados entre 28 e 30 objetos nas simulações em que as ações corretivas são utilizadas e 28 objetos quando estas ações estão desativadas. Em todas as dez simulações que utilizam ações corretivas, conforme a Tabela B.1, a Tarefa 1 executa 82 iterações e cumpre o *deadline* 55 vezes (67,07%), sendo que 17 (30,9%) ocorrem por consequência do uso de ações corretivas. As restrições temporais da Tarefa 1, nas simulações em que não há a utilização de ações corretivas, são respeitadas em 44 (62,85%) de suas 70 iterações. O tempo total que os robôs levam para completar as tarefas a cada iteração e o *deadline* das tarefas são comparados na Figura B.1. As comparações realizadas nesta figura apenas dizem respeito aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial do histórico de 100 entradas.

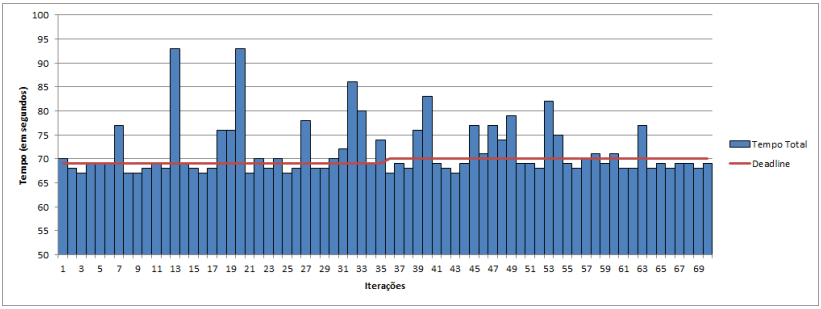
Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	82	70
Cumpre *Deadline*	55 (67,07%)	44 (62,85%)
Não Cumpre *Deadline*	27 (32,93%)	26 (37,15%)
Cumpre *DL* c/ A. C.	17	-
Cumpre *DL* s/ A. C.	38	44

Tabela B.1: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 1

A Tabela B.2 também apresenta uma comparação entre os va-



(a) Com ações corretivas

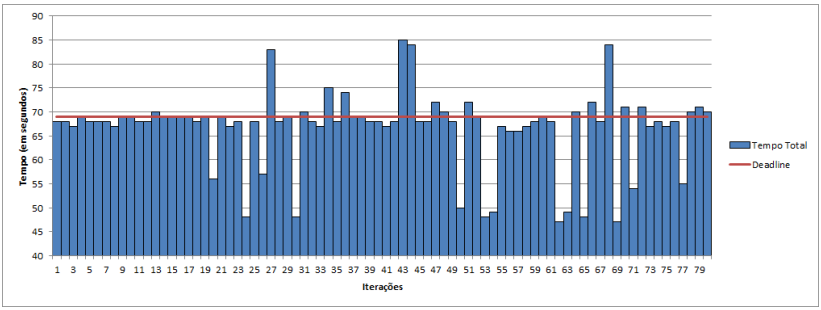


(b) Sem ações corretivas

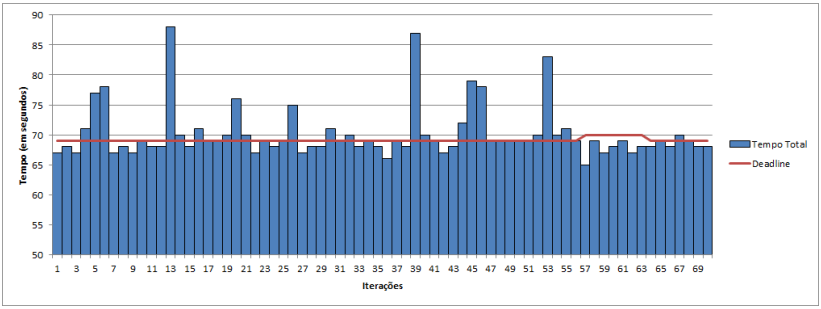
Figura B.1: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 1

lores obtidos para a Tarefa 2 em ambas as abordagens. Esta tarefa é executada 80 vezes quando as ações corretivas estão habilitadas e 70 vezes quando elas não são utilizadas. Das 80 iterações da primeira abordagem, 62 (77,5%) conseguem cumprir o *deadline* e as ações corretivas são responsáveis por 14 (22,58%) dessas 80 iterações. Na segunda abordagem, 47 (67,14%) iterações são capazes de respeitar as restrições temporais da tarefa. Na Figura B.2 são apresentados gráficos comparando o tempo que os robôs levam para cumprir as tarefas e o *deadline* cada iteração. Esta Figura compara os resultados obtidos para a Tarefa 2 nas duas abordagens e com um tamanho inicial de histórico igual a 100 entradas.

O valor do *deadline* da Tarefa 1 varia entre 69 e 70 segundos nas simulações em que as ações corretivas são utilizadas, enquanto que o da Tarefa 2 se mantém em 69 segundos. Nestas simulações, a média de objetos carregados é de 13,4 e de 14,5 para a Tarefa 1 e para a Tarefa 2,



(a) Com ações corretivas



(b) Sem ações corretivas

Figura B.2: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 1

respectivamente. Nas simulações realizadas sem as ações corretivas, a variação do valor do *deadline* se mantém no intervalo fechado entre 69 e 70 segundos em ambas as tarefas. A média de objetos carregados por simulação é de 14 objetos para as duas tarefas. A quantidade de objetos carregados em ambas as abordagens, no Mapa 1, é comparada, para as tarefas 1 e 2, respectivamente, nos gráficos apresentados nas Figuras B.3a e B.3b. Estas figuras apresentam comparações para as simulações em que o tamanho inicial de histórico é igual a 100 entradas.

B.2. Mapa 2

A quantidade de objetos carregados em cada uma das dez simulações em que o mecanismo é utilizado varia entre 36 e 39 objetos. Nas simulações em que as ações corretivas não são utilizadas, carregou-se entre 34 e 36 objetos. A Tabela B.3 compara os resultados obtidos

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	80	70
Cumpre *Deadline*	62 (77,5%)	47 (67,14%)
Não Cumpre *Deadline*	18 (22,5%)	23 (32,86%)
Cumpre *DL* c/ A. C.	14	-
Cumpre *DL* s/ A. C.	48	47

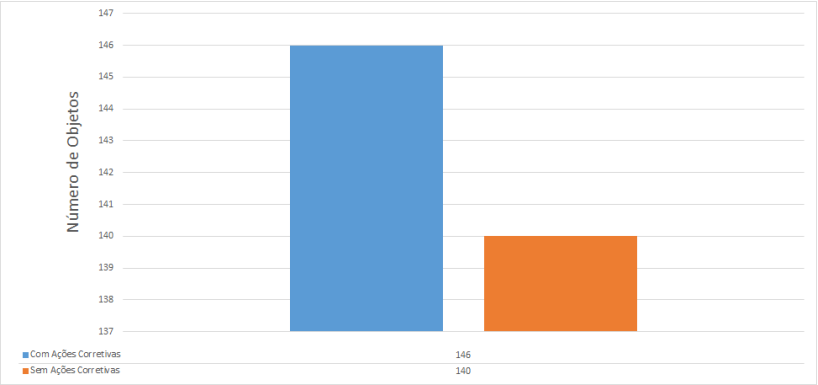
Tabela B.2: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 1

para a Tarefa 1 nas duas abordagens. Juntando as dez simulações, utilizando ações corretivas, a Tarefa 1 é executada 114 vezes, dentre elas, 74 (64,91%) são capazes de cumprir o *deadline* e 16 (21,62%), destas 74, são devido à utilização de ações corretivas. Sem estas ações, a Tarefa 1 é executada 102 vezes e seu *deadline* é cumprido 67 vezes (65,68%). A Figura B.4 mostra uma comparação entre o tempo total que os robôs levam para completar as tarefas e o *deadline* de cada tarefa. A comparação da Figura B.4 se refere aos resultados obtidos para a Tarefa 1 em ambas as abordagens e com o tamanho inicial de histórico igual a 100 entradas.

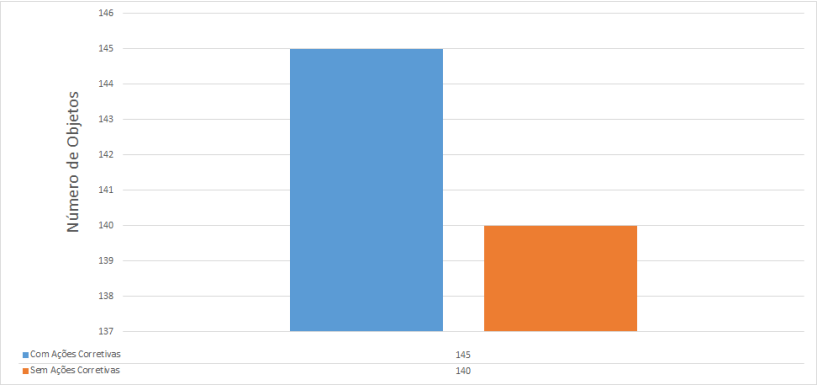
Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	114	102
Cumpre *Deadline*	74 (64,91%)	67 (65,68%)
Não Cumpre *Deadline*	40 (35,09%)	35 (34,32%)
Cumpre *DL* c/ A. C.	16	-
Cumpre *DL* s/ A. C.	58	67

Tabela B.3: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 2

Conforme os resultados apresentados na Tabela B.4, nas simulações em que há a utilização de ações corretivas, na Tarefa 2, são executadas 106 iterações e 52 (49,05%) cumprem o *deadline*. Dentre estas 52 iterações, 51 (98,07%) ocorrem devido ao uso das ações corretivas. Sem estas ações, a Tarefa 2 é executada 73 vezes e cumpre o *deadline* em 42 (57,53%) dessas 73 iterações. Uma comparação entre o tempo total para completar uma tarefa e o *deadline* da tarefa é apresentada na Figura B.5. Esta comparação se refere aos resultados obtidos para a Tarefa 2 em ambas as abordagens e com o tamanho inicial de histórico igual a 100 entradas.



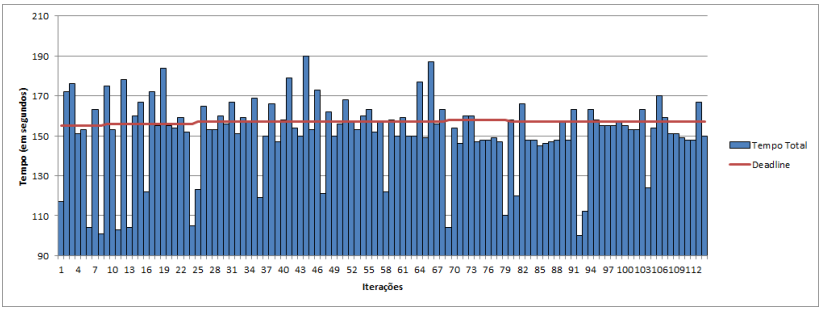
(a) Tarefa 1



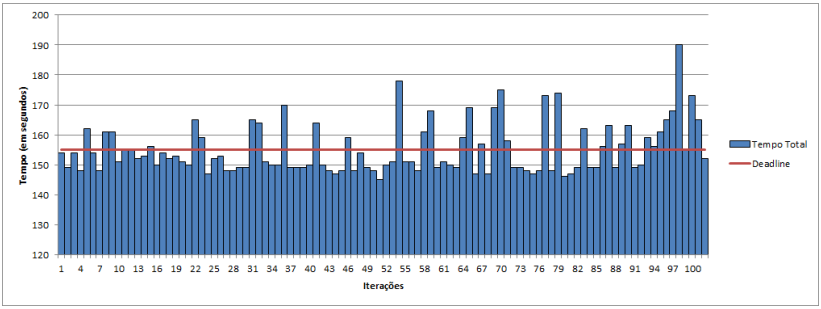
(b) Tarefa 2

Figura B.3: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 1

Dentre as simulações que utilizam ações corretivas, o valor do *deadline* varia entre 155 e 158 segundos na Tarefa 1 e entre 202 e 210 segundos na Tarefa 2. A média de objetos carregados por simulação é de 21 objetos para a Tarefa 1 e 16,1 objetos para a Tarefa 2. Nas simulações que não utilizam ações corretivas o *deadline* se mantém em 155 segundos na Tarefa 1 e varia entre 210 e 211 segundos na Tarefa 2. A média de objetos carregados por simulação é de 20,4 objetos para a Tarefa 1 e de 14,6 objetos para a Tarefa 2. As Figuras B.6a e B.6b comparam a quantidade de objetos carregados para as tarefas 1 e 2,



(a) Com ações corretivas



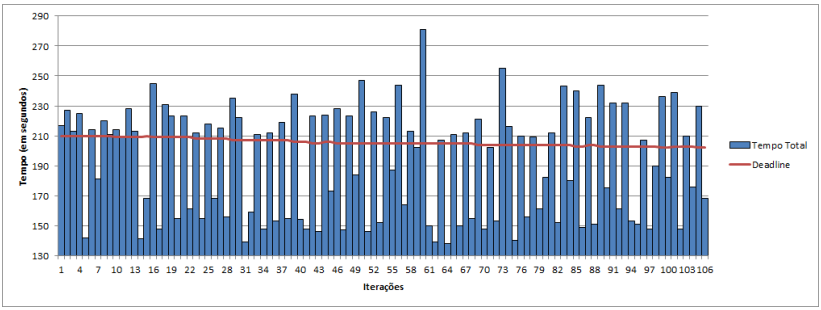
(b) Sem ações corretivas

Figura B.4: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 2

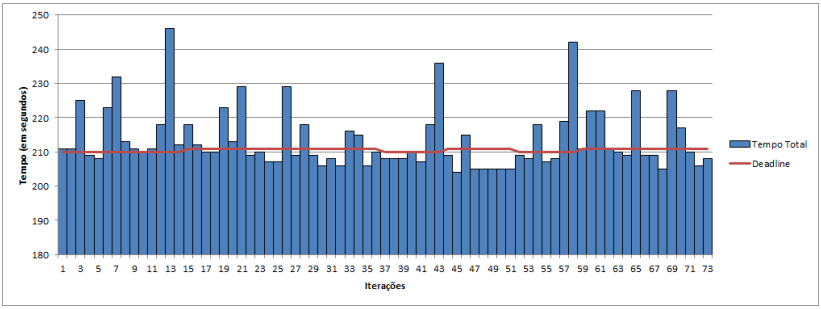
respectivamente, para o Mapa 2 e com tamanho inicial de histórico de 100 entradas.

B.3. Mapa 3

Dentre as dez simulações realizadas em que ações corretivas são utilizadas, a quantidade de objetos carregados está no intervalo fechado entre 13 e 14 objetos, ao passo que são carregados 12 objetos nas simulações que as ações corretivas não são utilizadas. Como apresentado na Tabela B.5, que compara os valores obtidos para a Tarefa 1 nas duas abordagens, dentre todas as dez simulações nas quais as ações corretivas estão ativadas, a Tarefa 1 é executada 42 vezes e cumpre o *deadline* da tarefa em 23 (54,76%) iterações. Destas 23, 14 (60,86%) são devido à utilização de ações corretivas. Quando estas ações estão desabilitadas, a Tarefa 1 é executada 30 vezes e consegue respeitar as



(a) Com ações corretivas



(b) Sem ações corretivas

Figura B.5: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 2

restrições temporais da tarefa 16 vezes (53,34%). Os gráficos da Figura B.7 apresentam uma comparação entre o tempo total que os robôs levam para concluir as tarefas e o *deadline* de cada tarefa. Esta Figura compara os resultados obtidos na Tarefa 1 em ambas as abordagens e tendo um histórico com 100 entradas como histórico inicial. A Tabela B.5 compara os valores obtidos para a Tarefa 1 nas duas abordagens.

Quando as ações corretivas estão sendo utilizadas, a Tarefa 2 executa 42 iterações e o *deadline* é cumprido em 26 (61,9%) delas, como apresentado na Tabela B.6, que compara os resultados da Tarefa 2 em ambas as abordagens. Destas 26 iterações, 15 (57,69%) conseguem cumprir o *deadline* devido ao uso das ações corretivas. Desabilitando estas ações, a Tarefa 2 é executada 30 vezes e cumpre o *deadline* em 15 (50%) dessas 30 iterações. Na Figura B.8, é apresentada, por meio de gráficos, uma comparação entre o valor do *deadline* de cada tarefa e o tempo que os robôs levam pra concluí-las em cada iteração. As

Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	106	73
Cumpre *Deadline*	52 (49,05%)	42 (57,53%)
Não Cumpre *Deadline*	54 (50,95%)	31 (42,47%)
Cumpre *DL* c/ A. C.	51	-
Cumpre *DL* s/ A. C.	1	42

Tabela B.4: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 2

Tarefa 1	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	42	30
Cumpre *Deadline*	23 (54,76%)	16 (53,34%)
Não Cumpre *Deadline*	19 (45,24%)	14 (46,66%)
Cumpre *DL* c/ A. C.	14	-
Cumpre *DL* s/ A. C.	9	16

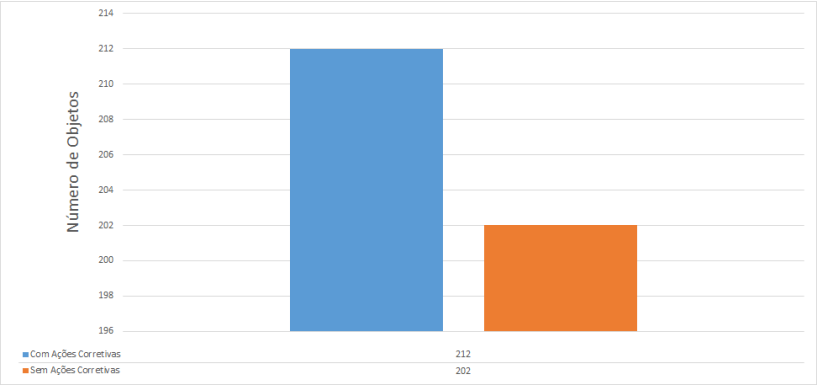
Tabela B.5: Tabela comparando os resultados obtidos na Tarefa 1 em ambas as abordagens no Mapa 3

comparações da Figura B.8 se referem aos resultados obtidos para a Tarefa 2 em ambas as abordagens com o tamanho inicial de histórico igual a 100 entradas.

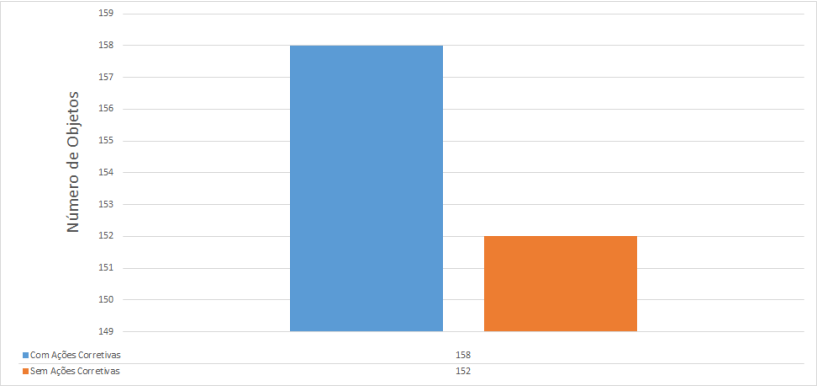
Tarefa 2	A. C. Habilitadas	A. C. Desabilitadas
Nº de Iterações	42	30
Cumpre *Deadline*	26 (61,9%)	15 (50%)
Não Cumpre *Deadline*	16 (38,1%)	15 (50%)
Cumpre *DL* c/ A. C.	15	-
Cumpre *DL* s/ A. C.	11	15

Tabela B.6: Tabela comparando os resultados obtidos na Tarefa 2 em ambas as abordagens no Mapa 3

A variação no *deadline* das duas tarefas em simulações que utilizavam ações corretivas assume valores entre 492 e 496 segundos na Tarefa 1 e entre 485 e 484 segundos na Tarefa 2. Quando as ações corretivas não estão sendo utilizadas, o valor do *deadline* varia entre 492 e 496 segundos na Tarefa 1 e entre 485 e 488 segundos na Tarefa 2. A média de objetos carregados por simulação apresenta resultados diferentes em cada abordagem, sendo que, para as simulações que utili-



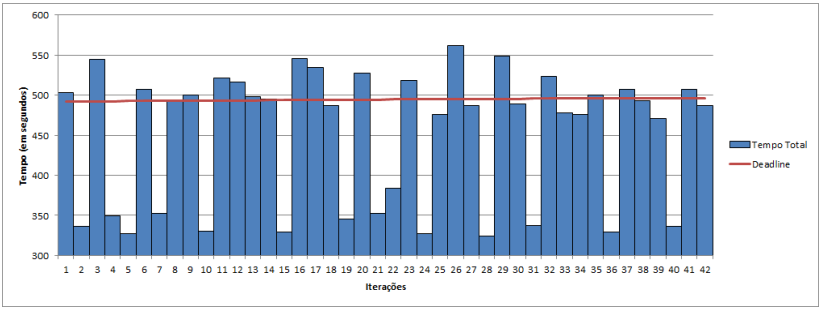
(a) Tarefa 1



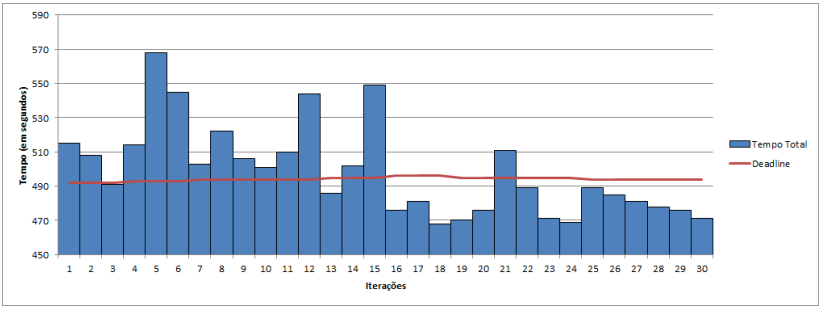
(b) Tarefa 2

Figura B.6: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 2

zam ações corretivas, essa média é de 7 objetos para a Tarefa 1 e de 6,9 objetos para a Tarefa 2. Nas simulações que as ações corretivas estão desativadas, a média de objetos carregados é de 6 objetos para ambas as tarefas. Uma comparação entre os objetos carregados no Mapa 3, com tamanho inicial de histórico de 50 entradas, é apresentada nas Figuras B.9a e B.9b para as tarefas 1 e 2 respectivamente.

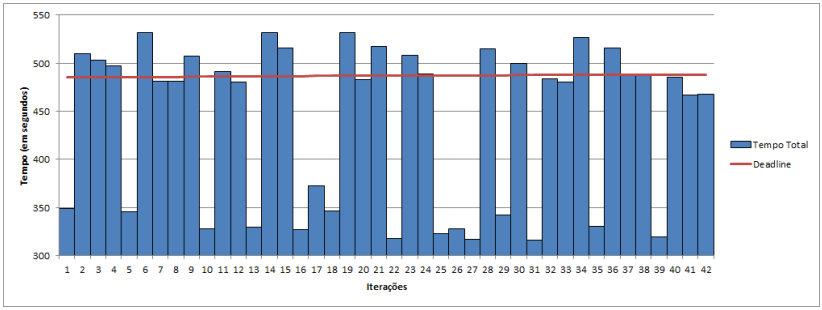


(a) Com ações corretivas

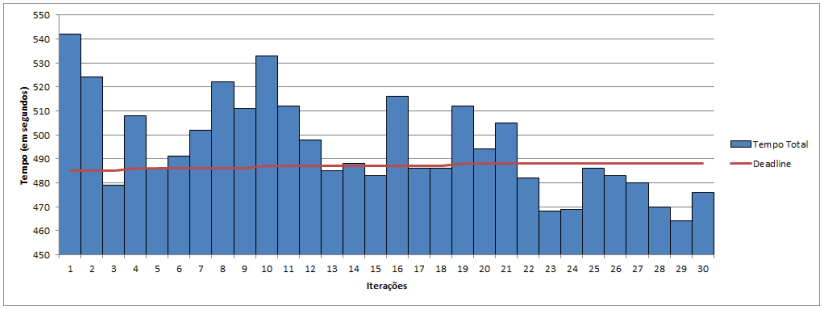


(b) Sem ações corretivas

Figura B.7: Gráficos comparando os resultados obtidos para a Tarefa 1 em ambas as abordagens no Mapa 3

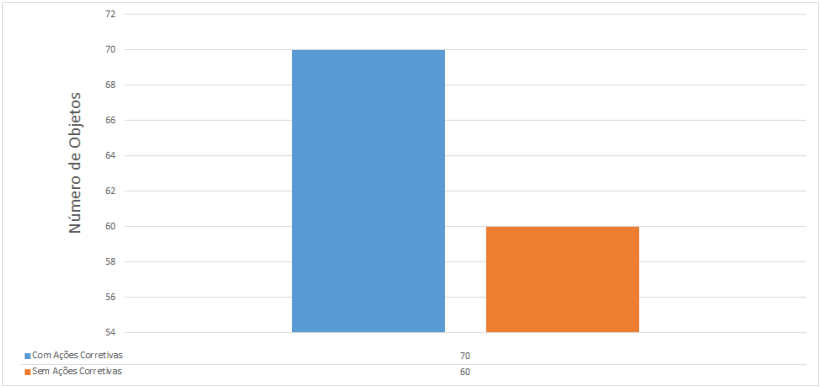


(a) Com ações corretivas

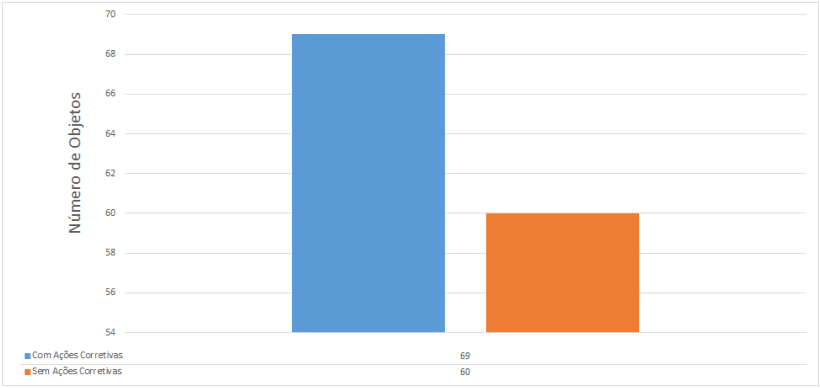


(b) Sem ações corretivas

Figura B.8: Gráficos comparando os resultados obtidos para a Tarefa 2 em ambas as abordagens no Mapa 3



(a) Tarefa 1



(b) Tarefa 2

Figura B.9: Gráficos apresentando a quantidade de objetos carregados nas dez simulações para a Tarefa 1 em ambas as abordagens no Mapa 3